I^2C over DS90UB913/4 FPD-Link III with Bidirectional Control Channel

Mahendra Patel

ABSTRACT

This application note describes I^2C communication between DS90UB913/914 devices through the FPD-Link III bidirectional control channel (referred as BCC). The low latency BCC interface allows the master I^2C device to remotely communicate with peripherals across the serial link.

Contents

1 Introduction

The DS90UB913/914 chipsets support full-duplex transmission of data with an embedded BCC over a single differential link. The BCC interface is I^{2}C compatible according to the I^{2}C standard. It provides access to programmable functions and registers on the local and remote device(s).

Three types of operations are supported for I^{2}C transactions over the bidirectional control channel, as shown in Figure 1,

1. Local (between Local I^{2}C and Host I^{2}C device)
2. Remote (between Remote SER and Host I^{2}C device)
3. Remote slave (between Remote Slave and Host I^{2}C device)

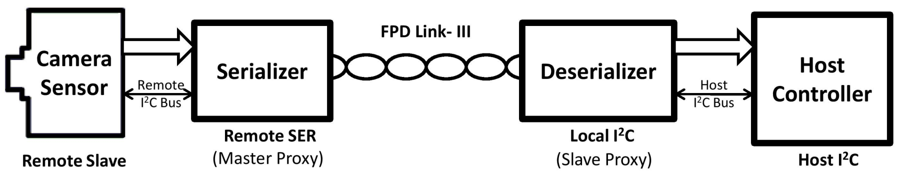

Figure 1. Typical FPD-Link III Connection with I^{2}C Bus

Each device can function as an I^{2}C slave proxy or master proxy depending on which side of the serial link I^{2}C Host controller is present.

- While addressing a remote peripheral or Ser/Des, the slave proxy (device connected to the host I^{2}C bus) will forward any byte transactions sent by the Host controller to the target device.
- Other device (Ser or Des connected to the remote I^{2}C bus) will function as a master proxy device, i.e., it acts as a master on behalf of the I^{2}C host controller. SCL frequency of the master proxy is register programmable.

The Ser/Des interface acts as a virtual bridge between host controller and the remote devices.

Local operations use standard master to slave operations to communicate with the local Serializer or Deserializer. Local I^{2}C operations do not result in transactions across the bidirectional control link and thus do not require any clock stretching by the slave. However, in order to communicate with remote devices attached to the remote I^{2}C bus, slave clock stretching must be supported by the I^{2}C host controller. The DS90UB913/914 chipset employ I^{2}C clock stretching during remote data transmission. Note that the slave device does not control the clock but only stretches it (by holding it low) until the remote peripheral has responded.
Refer to application report SNLA131A / AN-2173 for more details.

Generally used I^{2}C Adapters are Aardvark, Corelis, iPort, etc.

2 Basic Definitions and I2C Control Registers

Device ID—Refers to the 7-bit physical I^{2}C address of the device. This report will use terms like DES ID, SER ID ans Slave ID, which means 7-bit physical I^{2}C address of deserializer, serializer and remote slave respectively.

Device Alias—Refers to the alternate 7-bit address assigned to either Serializer or Deserializer or remote slave. Device Alias helps to differentiate between the devices having same Device ID or physical I^{2}C address. It is recommended that the I^{2}C master should always use the device alias to communicate with a remote I^{2}C slave.

NOTE: I^{2}C addresses are always 7bits (binary). For DS90UB913/914, majority of the registers associated with I^{2}C addresses uses 'bits [7:1]' for address and 'bit 0' is either reserved or used for some other purpose. Hence, while loading address value to a specific register, it is always left shifted by 1bit.
E.g., 0x50 (101 0000) left shifted by 1 bit is 0xA0 (1010 0000).
This operation can be represented as, **0x50<<1** which is equal to **0xA0**.

Pass Through— This I^2C control enables communication with all the remote devices, whose device ID and corresponding device Alias fields is defined into the local device (Ser or Des), as discussed in Section 3.

Pass Through All— This control enables all I^2C transaction over the serial link that are not addressing the local device. Refer Table 3 and Table 5.

Table 1 lists basic controls, which define the scope of I^2C bidirectional control in a system using single or multiple DS90UB913/914 serial links:

Table 1. I^2C Control Registers

I^2C Control	DS90UB914 registers / bits (Hex)		DS90UB913 registers / bits (Hex)	
	Address	Default Value	Address	Default Value
I^2C Device ID	0x00	0xC0	0x00	0xB0
SER ID	0x06	Auto loaded from Ser	Not applicable	Not applicable
SER Alias	0x07	0x00		
DES ID	Not applicable	Not applicable	0x06	Auto loaded from Des
DES Alias			0x07	0x00
Slave ID	0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F	0x00	0x08	0x00
Slave Alias	0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17	0x00	0x09	0x00
SCL High Time	0x40	0x82	0x11	0x82
SCL Low Time	0x41	0x82	0x12	0x82
I^2C Pass Through (bit)	0x03[3]	1	0x03[2]	1
I^2C Pass Through All (bit)	0x21[7]	0	0x03[3]	0

NOTE: I^2C Device ID defined by register setting is overridden by the ID[x] pin setting, refer Table 2 and device datasheet for more details

Table 2. I²C Control Register Description

Name	Bits	Field	R/W	Description
I²C Device ID	7:1	DEVICE ID	R/W	7-bit address I²C device address. 0x58 (101 1000) is the default value for DS90UB913 and 0x60 (110 0000) is the default value for DS90UB914.
	0	SER ID SEL		0: Device ID is set from ID[x], which is default. 1: Register I²C Device ID[7:1] overrides address selected by ID[x]
SER ID	7:1	Remote ID	RW	7-bit Serializer Device ID configures the I²C Slave ID of the remote Serializer. A value of 0 in this field disables I²C access to the remote Serializer. This field is automatically configured by the Bidirectional Control Channel once RX Lock has been detected. Software may overwrite this value, but should also assert the FREEZE DEVICE ID bit to prevent overwriting by the Bidirectional Control Channel.
	0	Freeze Device ID		When set to '1', it prevents auto loading of Serializer Device ID from the Forward Channel. The ID will be frozen at the value written in bit[7:1] of this register. The default setting is '0'.
SER Alias	7:1	Serializer Alias ID	RW	7-bit Remote Serializer Device Alias ID configures the decoder for detecting transactions designated for an I²C Serializer device. The transaction will be re-mapped to the address specified in the SER ID register. A value of 0 in this field disables access to the remote I²C SER, unless I²C pass through all is enabled on the local DES.
	0	RSVD		Reserved
DES ID	7:1	Deserializer Device ID	RW	7-bit Deserializer Device ID configures the I²C Slave ID of the remote Deserializer. A value of 0 in this field disables I²C access to the remote Deserializer. This field is automatically configured by the Bidirectional Control Channel once RX Lock has been detected. Software may overwrite this value, but should also assert the FREEZE DEVICE ID bit to prevent overwriting by the Bidirectional Control Channel.
	0	Freeze Device ID		When set to '1', it prevents auto loading of Deserializer Device ID by BCC. The ID will be frozen at the value written in bit[7:1] of this register. The default setting is '0'.
DES Alias	7:1	Deserializer Alias ID	RW	7-bit Remote Deserializer Device Alias ID configures the decoder for detecting transactions designated for an I²C Deserializer device. The transaction will be re-mapped to the address specified in the DES ID register. A value of 0 in this field disables access to the remote I²C DES, unless I²C pass through all is enabled on the local SER.
	0	RSVD		Reserved
Slave ID	7:1	Slave ID	RW	7-bit Remote Slave Device ID configures the physical I²C address of the remote I²C Slave device attached to the remote Ser/Des. If an I²C transaction is addressed to the Slave Alias ID, the transaction will be re-mapped to this address before passing the transaction across the Bidirectional Control Channel. A value of 0 in this field disables access to the remote I²C slave.
	0	RSVD		Reserved
Slave Alias	7:1	Slave Alias ID	RW	7-bit Remote Slave Device Alias ID configures the decoder for detecting transactions designated for an I²C Slave device attached to the remote Ser/Des. The transaction will be re-mapped to the address specified in the Slave ID register. A value of 0 in this field disables access to the remote I²C Slave.
	0	RSVD		Reserved
SCL High Time	7:0	SCL High Time	RW	This field configures the high pulse width of the SCL output when the device is acting as a proxy master. Units are 50ns for the nominal oscillator clock frequency and the default value is 0x82 which corresponds to 6.5µs.
SCL Low Time	7:0	SCL Low Time	RW	This field configures the low pulse width of the SCL output when the device is acting as a proxy master. This value is also used as the SDA setup time by the I²C slave for providing data prior to releasing SCL during accesses over the BCC. Units are 50ns for the nominal oscillator clock frequency and the default value is 0x82 which corresponds to 6.5µs.

3 I²C Bidirectional Control Channel : Operation Examples

This section describes various example cases to communicate with a particular device within a system and corresponding register settings required. Depending upon which side of serial link the host I²C controller is located, examples are classified in two major categories.

3.1 I²C Master attached to Deserializer

In this configuration, DS90UB914 acts as a slave on the local I²C bus and DS90UB913 will act as a master proxy on the remote I²C bus, refer Figure 1. On DS90UB914, eight Slave IDs and corresponding Slave Alias can be defined, which allows I²C communication with maximum of eight remote slaves attached to Serializer.

NOTE: All the register settings in this section will refer to DS90UB914 only.

1. **Local I²C**
   In this case, communication between Host controller and attached Deserializer do not require any explicit action to be taken. Once the Deserializer is powered up, I²C commands can be exchanged between the two, using I²C device ID (0x60 for this example) of the Deserializer, which is set by IDx pin by default.

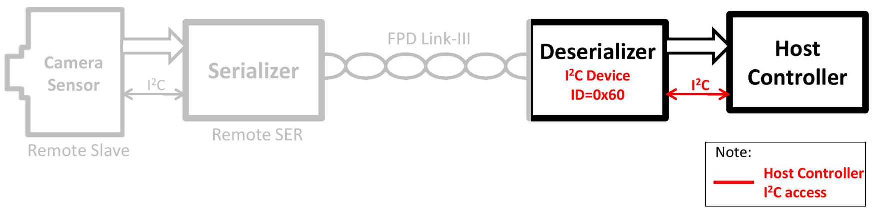

Figure 2. Local I²C

2. **Communication with remote SER**
   This mode requires additional settings to pass the I²C commands over the Serial Link. Figure 3 shows an example to communicate with Remote SER.
   Method 1:
   Content of reg(0x06), SER ID is auto loaded from Serializer, once the serial link is established. The content will be 0xB0 in this case (0x58<<1). It is necessary to define reg(0x07), SER Alias (other than 0x00, which is default) to allow passage of I²C command intended for Remote SER. In this example, it has been assigned a value of 0xB2 (0x59<<1). Hence, Host controller can communicate to Remote SER using SER Alias 0x59.
   Method 2:
   Alternatively, setting bit 0x21[7] to 1 allows communication with Remote SER using its I²C Device ID, i.e.0x58 itself. However, in this setting, all the I²C Device IDs that do not match with Deserializer I²C device ID are re-mapped to Remote SER.

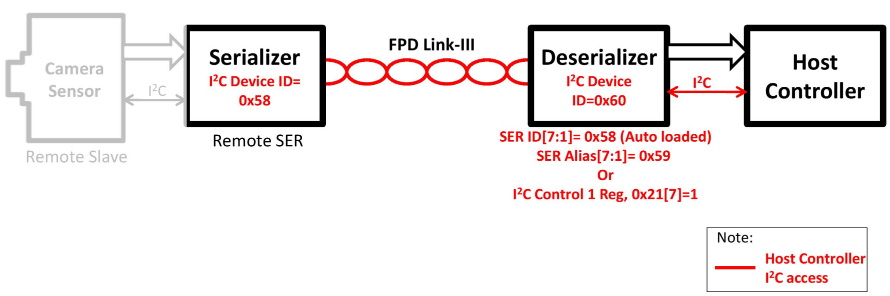

Figure 3. Communication with remote SER

3. **Communication with Remote slave/Camera attached to Remote SER**
   To communicate with a Remote Slave, like a camera sensor in Figure 4, it is required to initialize Slave ID[X] and Slave Alias[X].

In the example stated, camera sensor has physical I^2C address of 0x50. Hence, initialize reg(0x08), Slave ID[0] as 0xA0 (0x50<<1) and initialize reg(0x10), Slave Alias [0] as 0xA2 (0x51<<1) for instance. Any commands initiated by host controller with address 0x51 are re-mapped to camera sensor (0x50) by Deserializer. Slave Alias can be same as Slave ID, if there are no duplicate addresses.

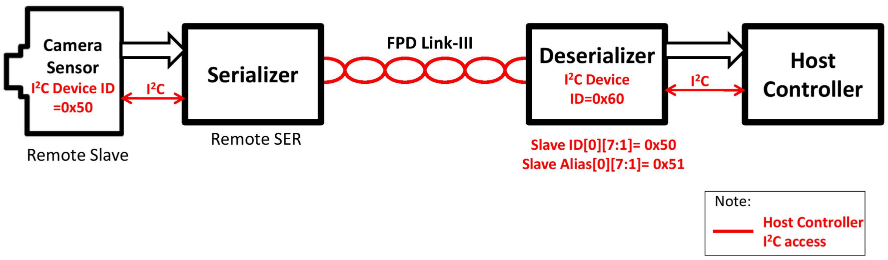

Figure 4. Communication with Remote slave/Camera attached to Remote SER

4. **Addressing Multiple Remote SER with duplicate device ID**
   In the Figure 5 shown below, there are two serial links and both the Serializers have identical device IDs, for e.g., 0x58. Note that Deserializers should have different I^2C device IDs to initialize and to communicate with them individually. Reg(0x06), SER ID in both the Deserializers are auto loaded as 0xB0 (0x58<<1) from the respective Serializers.
   To distinctly communicate with each of the Serializer, assign different values to reg(0x07), SER Alias of both Deserializers. For example, write 0xAE (0x57<<1) into reg(0x07), the SER Alias of Deserializer1 and 0xB2 (0x59<<1) into reg(0x07), the SER Alias of Deserializer2. This will allow communication with Serializer1 using I^2C address 0x57 from Host controller and also, with Serializer2 using I^2C address 0x59.

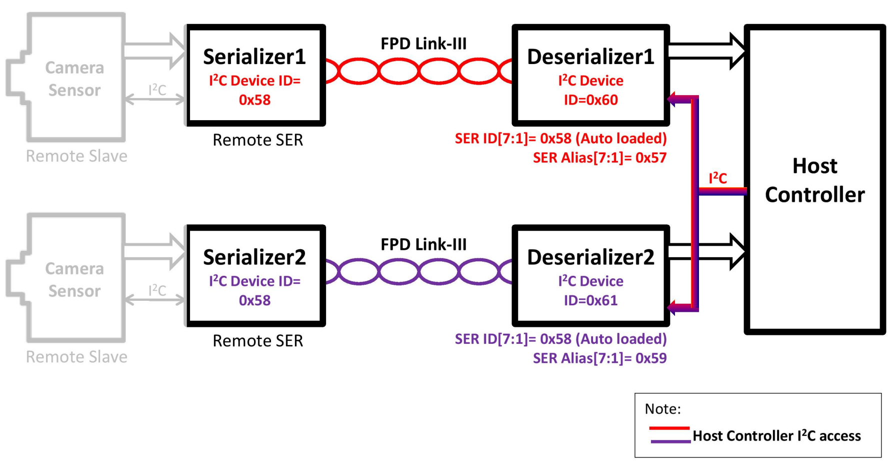

Figure 5. Addressing Multiple Remote SER with duplicate device ID

5. **Addressing Multiple Remote Slaves/Cameras with duplicate device ID**
   In a system with multiple serial links and Remote slaves having identical I^2C device ID, it is necessary to initialize the Slave ID[X] and Slave Alias[X] of each Deserializer. Note that Deserializers should have different I^2C device IDs to initialize and to communicate with them individually.
   In the example stated, each Camera Sensor has physical I^2C address of 0x50. Hence, we need to initialize reg(0x08), Slave ID[0] as 0xA0 (0x50<<1) for both Deserializers. However, reg(0x10), Slave Alias[0] should be assigned different values. For instance, write values 0xA2 (0x51<<1) into reg(0x10), Slave Alias[0] of Deserializer1 and 0xA4 (0x52<<1) into reg(0x10), Slave Alias[0] of Deserializer2. Any commands initiated by host controller with I^2C address 0x51 are re-mapped to Camera Sensor1 by Deserializer1. Deserializer2 does the same with all the commands having I^2C address 0x52 to allow communication with Camera Sensor2.

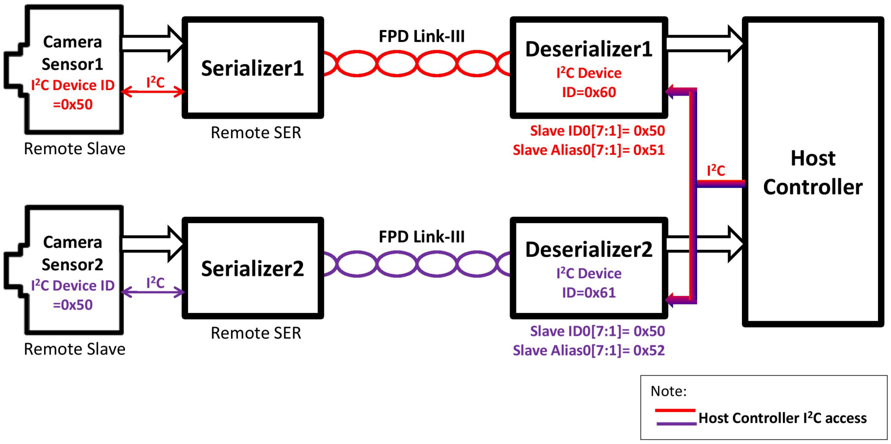

Figure 6. Addressing Multiple Remote Slaves/Cameras with duplicate device ID

Table 3 provides information on I^2C access to various devices in the system using "Pass Through" and "Pass Through All" control present on DS90UB914.

Table 3. Operation with "Pass Through" and "Pass Through All" : on DS90UB914

I^2C Pass Through: Bit 0x03[3]	I^2C Pass Through All: Bit 0x21[7]	Communication with Remote Serializer		Communication with Remote Slave: With Slave ID & Alias defined
		Without SER Alias defined	With SER Alias defined	
1 (Default)	0 (Default)	No	Yes	Yes
1	1	Yes	Yes	No
0	1	Yes	Yes	No
0	0	No	No	No

NOTE:

- Communication with remote slave is not possible without defining Slave ID[X] & Slave Alias[X] registers.
- SER ID value is auto loaded over serial link, unless bit 0x06[0] is set to 1 in Deserializer.

3.2 I^2C Master attached to Serializer

In this configuration, DS90UB913 acts as a slave on the local I^2C bus and DS90UB914 will act as a master proxy on the remote I^2C bus, refer Figure 7.

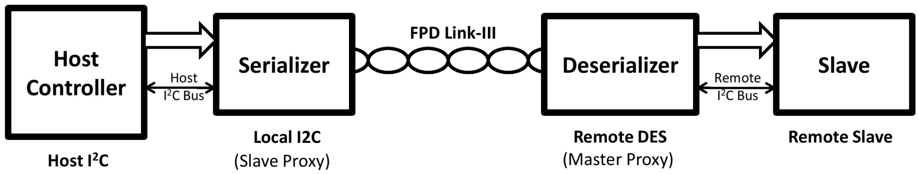

Figure 7. I^2C Master attached to Serializer

All modes listed below are applicable, when I^2C Master is attached to Serializer in the similar manner as discussed in Section 3.1.

1. Local I^2C
2. Communicating with remote DES
3. Communicating with Remote slave attached to DES
4. Addressing Multiple DES with duplicate device ID
5. Addressing Multiple slaves with duplicate device ID

Hence, a generalized example is discussed in this section.
The only difference is that, DS90UB913 has only one Slave ID and Slave Alias, as compared to 8 available on DS90UB914.

NOTE: All the register settings in this section refer to DS90UB913 only.

Generalized Example:
Figure 8 shows a system with two serial links having duplicate addresses for Remote DESes and Remote Slaves. Note that Serializers should have different I^2C device IDs to initialize and to communicate with them individually.

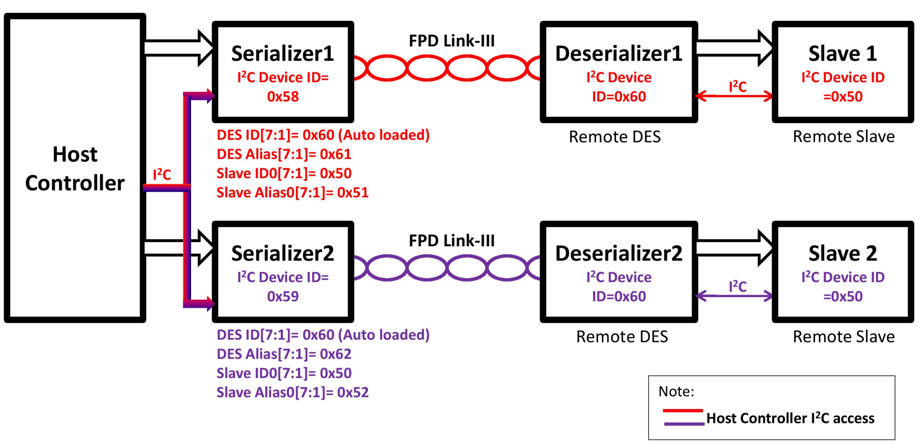

Figure 8. Addressing Multiple Remote Devices with duplicate device ID

Following Table 4 lists the steps to communicate with each of the device present in the system represented in Figure 8.

Table 4. Addressing Multiple Remote Devices with duplicate device ID

Step	Communication with	Action required				
		On Device (I^2C Address)	Reg (HEX addr.)	R / W	Value (HEX)	Comments
1	Deserializer1	Serializer1 (0x58)	DES ID (0x06)	Read	0xC0	Auto loaded from deserializer (0x60<<1)
			DES Alias (0x07)	Write	0xC2	Assign alias to refer Deserializer1 (0x61<<1)
		Deserializer1 (0x61)	Any	R/W	-	Use address 0x61 to communicate with Deserializer1, instead of 0x60
2	Deserializer2	Serializer2 (0x59)	DES ID (0x06)	Read	0xC0	Auto loaded from deserializer (0x60<<1)
			DES Alias (0x07)	Write	0xC4	Assign alias to refer Deserializer2 (0x62<<1)
		Deserializer2 (0x62)	Any	R/W	-	Use address 0x62 to communicate with Deserializer2, instead of 0x60
3	Slave 1	Serializer1 (0x58)	Slave ID (0x08)	Write	0xA0	Assign 0x50<<1, because physical I^2C address of slave is 0x50
			Slave Alias (0x09)	Write	0xA2	Assign alias to refer Slave1 (0x51<<1)
		Slave 1 (0x51)	Any	R/W	-	Use address 0x51 to communicate with Slave1, instead of 0x50
4	Slave 2	Serializer2 (0x59)	Slave ID (0x08)	Write	0xA0	Assign 0x50<<1, because physical I^2C address of slave is 0x50
			Slave Alias (0x09)	Write	0xA4	Assign alias to refer Slave2(0x52<<1)
		Slave 2 (0x52)	Any	R/W	-	Use address 0x52 to communicate with Slave2, instead of 0x50

Summarizing the operation,

- Any I^2C commands with address 0x61 are re-mapped to Deserializer1
  AND
  Commands with I^2C address 0x51 are re-mapped to Slave1 by Serializer1.
- Any I^2C commands with address 0x62 are re-mapped to Deserializer2
  AND
  Commands with I^2C address 0x52 are re-mapped to Slave2 by Serializer2.

Table 5 provides information on I^2C access to various devices in the system using "Pass Through" and "Pass Through All" control present on DS90UB913.

Table 5. Operation with "Pass Through" and "Pass Through All" : on DS90UB913

I^2C Pass Through: Bit 0x03[2]	I^2C Pass Through All: Bit 0x03[3]	Communication with Remote Deserializer		Communication with Remote Slave: With Slave ID & Alias defined
		Without DES Alias defined	With DES Alias defined	
1 (Default)	0 (Default)	No	Yes	Yes
1	1	Yes	Yes	No
0	1	Yes	Yes	No
0	0	No	No	No

NOTE:

- Communication with remote slave is not possible without defining Slave ID & Slave Alias registers.
- DES ID value is auto loaded over serial link, unless bit 0x06[0] is set to 1 on Serializer.

4 Configuration of SCL on the Proxy Master

Referring to Table 2, by writing the appropriate value in registers **"SCL High Time"** and **"SCL Low Time"** of master proxy, SCL frequency can be configured for the remote I²C bus. By default, this registers are set to 0x82 i.e., 130 (decimal). Increasing or decreasing this value by 1 will change the SCL high / low time by 50ns, as unit step size is 50ns.
Therefore,
SCL high time = 130 × 50ns = 6.5µs and SCL low time = 130 × 50ns = 6.5µs, for nominal oscillator frequency.
For this register settings, SCL period would be 13µs (6.5µs + 6.5µs) and typical SCL frequency would be 77kHz.

Table 6 lists the register settings for 100kHz and 400kHz SCL frequency also. Minimum recommended SCL low time is 1.3µs and SCL high time is 0.6µs as per the device datasheet. In addition to this specification, it is important to consider setup and hold time requirement of the remote slave device, while deciding the SCL frequency.

While communicating to a remote device over the serial link, each I²C data byte is buffered and regenerated on the remote side of the link, hence the overall I²C throughput will be reduced. The reduction is dependent on the operating frequencies of the local and remote interfaces. The local I²C rate is based on the host controller clock rate, while the remote rate (SCL frequency) depends on the register settings of the I²C master proxy.

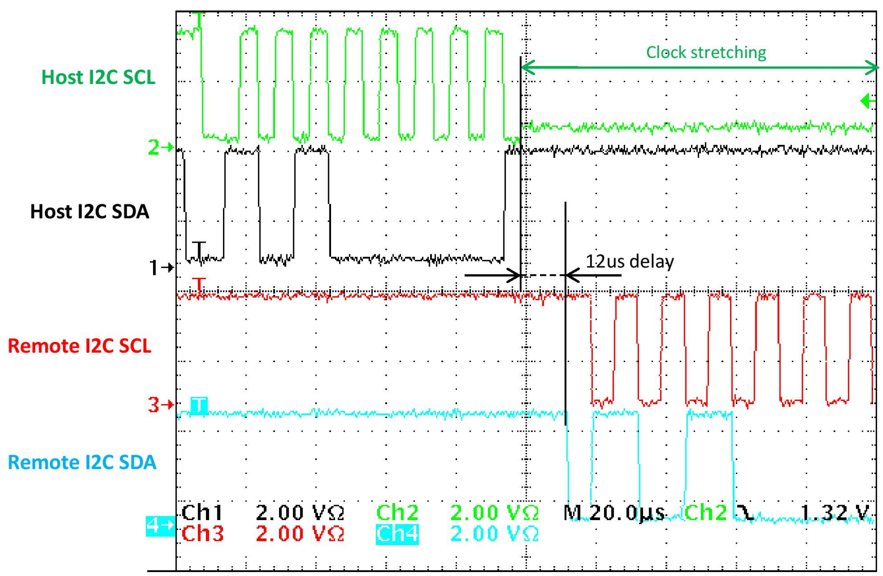

Figure 9. BCC Delay

DS90UB913/914 chipset communicates I²C commands over the back channel in a form of 30bit frames and the frame rate is 83.33kHz. Data over the back channel is buffered and sent over the serial link at the interval of 12µs (1/83.33kHz). Hence, typical BCC delay is of 12us, however it may extend up to maximum of 24us. Figure 9 shows a scope shot of this delay.

5 Data Throughput To Remote I²C Slaves

For purpose of understanding effects of the BCC delay on data throughput from a host controller to a remote I²C device (master proxy), the approximate bit rate including latency timings across the control channel can be calculated as,
9 bits / ((Host_bit * 9) + (Remote_bit * 9) + FCdelay + BCCdelay).
where, FCdelay means forward channel delay
Example: For 100kbit/s,
Host_bit = 1 / (100 kbit/s) = 10us
Remote_bit = 1 / (77 kHz) = 13us

FCdelay = 1us (max)
BCCdelay = 12us
Effective rate = 9bits / (90us + 117us + 1us + 12us) = 41kbit/s.

Since the I²C protocol includes overhead for sending address information as well as START and STOP bits, the actual data throughput depends on the size and type of transactions used. Use of large bursts to read and write data will result in higher data transfer rates. Following Table 6 lists typical I²C bit rates for DS90UB913/914 chipset and corresponding register settings required.

Table 6. Typical I²C bit rates

Host I²C Rate	Remote I²C Rate	Master Proxy register settings		Net Bit Rate
		"SCL High Time" register	"SCL Low Time" register	
100 kbit/s	77 kbit/s (Default)	0x82 (Default)	0x82 (Default)	41 kbit/s
100 kbit/s	100 kbit/s	0x64	0x64	47 kbit/s
400 kbit/s	100 kbit/s	0x64	0x64	72 kbit/s
400 kbit/s	400 kbit/s	0x32	0x32	155 kbit/s

6 Conclusion

This application note provides an overview of the I²C bus along with details describing the I²C interface specific to DS90UB913, DS90UB914 and I²C peripherals.

7 References

1. Application report: I2C Communication over FPD-Link III with Bidirectional Control Channel, http://www.ti.com/lit/an/snla131a/snla131a.pdf
2. DS90UB913Q-Q1 datasheet, http://www.ti.com/lit/ds/symlink/ds90ub913q-q1.pdf
3. DS90UB914Q-Q1 datasheet, http://www.ti.com/lit/ds/symlink/ds90ub914q-q1.pdf