DOCUMENT NUMBER
S12IICV2/D

HCS12 Inter-Integrated Circuit(IIC) Block Guide V02.08

Original Release Date: 08 SEP 1999
Revised: Jun 3, 2004

8/16 Bit Division,TSPG
Motorola, Inc.

Revision History

Version Number	Revision Date	Effective Date	Author	Description of Changes
0.1	8-Sep-99		Vipin Agrawal, Puneet Goel	Original draft. Distributed only within Motorola
0.2	30-Sep-99		Puneet Goel	Minor corrections as suggested by Joachim Kruecken.
2.0	12-Feb-01		Gautam Kar, Gurdarshan Kalra	Reformatted for SRS v2.0
2.1	2-Mar-2001		Gurdarshan Kalra	Minor corrections as suggested by Jens Winkler
2.2	6-Mar-2001		Gurdarshan Kalra	Minor corrections as suggested by Jens Winkler
2.03	26-Mar-2001		Gurdarshan Kalra Jens Winkler	Minor updates in format
2.04	19-July-2001		Dirk Rowald	Document names have been added, Names and variable definitions have been hidden
2.05	7-Mar-2002		Stephen Zhou	Minor updates in format
2.06	18-Aug-2002		Stephen Zhou	Reformated for SRS3.0,and add examples for programing general use and some diagrams to make it more user friendly as suggested by Joachim
2.07	11-Apr-2003		Stephen Zhou	Clearly claim support 400kps; Add notes for TCF bit in Section 5.1.3 Correct Section 7 for IBIF is cleared by writing '1'
2.08	3-Jun-2004		Vicers Cai	Correct the wrong divider values for SDA Hold from IBC=$60 to IBC=$7F

Table of Contents

List of Figures

List of Tables

Preface

N/A.

Section 1 Introduction

1.1 Overview

The Inter-IC Bus (IIC or I2C) is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange between devices Being a two-wire device, the IIC Bus minimizes the need for large numbers of connections between devices, and eliminates the need for an address decoder.

This bus is suitable for applications requiring occasional communications over a short distance between a number of devices. It also provides flexibility, allowing additional devices to be connected to the bus for further expansion and system development.

The interface will operate at baud rates of up to 100kbps with maximum capacitive bus loading.With reduced bus slew rate, the device is capable of operating at higher baud rates, up to a maximum of [MCUbus]clock/20. The module can operate up to a baud rate of 400kbps provided the IIC bus slew rate is less than 100ns. The maximum communication interconnect length and the number of devices that can be connected to the bus are limited by a maximum bus capacitance of 400pF in all instances.

1.2 Features

The IIC module has the following key features:

- Compatible with I2C Bus standard
- Multi-master operation
- Software programmable for one of 256 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection

1.3 Modes of Operation

The IIC functions the same in normal, special, and emulation modes. It has two low power modes, wait and stop modes.

- Run Mode

  This is the basic mode of operation.
- Wait Mode

  IIC operation in wait mode can be configured. Depending on the state of internal bits, the IIC can operate normally when the CPU is in wait mode or the IIC clock generation can be turned off and the IIC module enters a power conservation state during wait mode. In the latter case, any transmission or reception in progress stops at wait mode entry.
- Stop Mode

  The IIC is inactive in stop mode for reduced power consumption. The STOP instruction does not affect IIC register states.

1.4 Block Diagram

The block diagram of the IIC module is shown in **Figure 1-1**

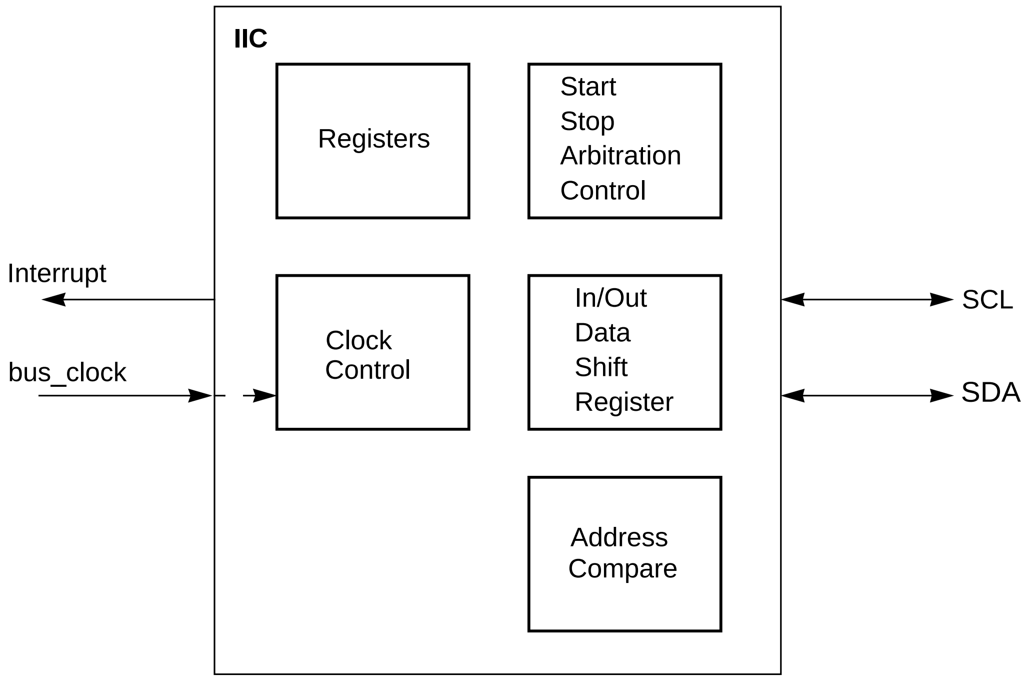
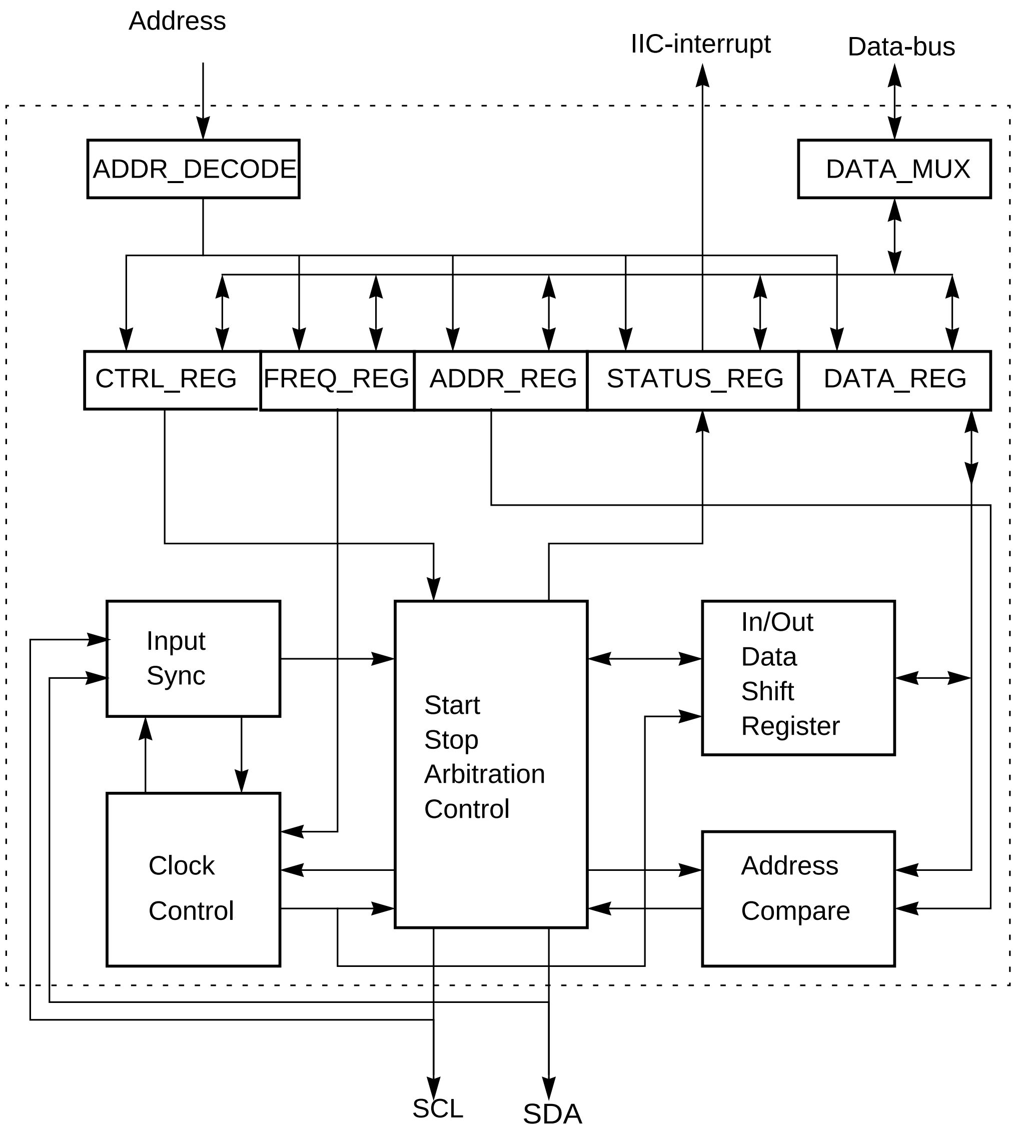


Figure 1-1 IIC Block Diagram


Address
IIC-interrupt
Data-bus
ADDR_DECODE
DATA_MUX
CTRL_REG
FREQ_REG
ADDR_REG
STATUS_REG
DATA_REG
Input Sync
Start Stop Arbitration Control
In/Out Data Shift Register
Clock Control
Address Compare
SCL
SDA

Section 2 External Signal Description

2.1 Overview

The IIC module has a total of 2 external pins.

2.2 Detailed Signal Descriptions

2.2.1 SCL

This is the bidirectional Serial Clock Line (SCL) of the module, compatible to the IIC-Bus specification.

2.2.2 SDA

This is the bidirectional Serial Data line (SDA) of the module, compatible to the IIC-Bus specification.

Section 3 Memory Map/Register Definition

3.1 Overview

This section provides a detailed description of all memory and registers for the IIC module.

3.2 Module Memory Map

The memory map for the IIC module is given below in **Table 3-1**. The Address listed for each register is the address offset.The total address for each register is the sum of the base address for the IIC module and the address offset for each register.

Table 3-1 Module Memory Map

Address	Use	Access
Base Address + $_0	IIC-Bus Address Register (IBAD)	Read/Write
Base Address + $_1	IIC-Bus Frequency Divider Register (IBFD)	Read/Write
Base Address + $_2	IIC-Bus Control Register (IBCR)	Read/Write
Base Address + $_3	IIC-Bus Status Register (IBSR)	Read/Write
Base Address + $_4	IIC-Bus Data I/O Register (IBDR)	Read/Write

3.3 Register Descriptions

This section consists of register descriptions in address order.Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

3.3.1 IIC Address Register

Register address: Base Address + $0000)

	7	6	5	4	3	2	1	0
R	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	0
W								▨
RESET:	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-1 IIC Bus Address Register (IBAD)

Read and write anytime{iic_regs}

This register contains the address the IIC Bus will respond to when addressed as a slave; note that it is not the address sent on the bus during the address transfer.{iic_slave}

ADR7–ADR1 — Slave Address

Bit 1 to bit 7 contain the specific slave address to be used by the IIC Bus module.{iic_slave}

The default mode of IIC Bus is slave mode for an address match on the bus.

RESERVED

Bit 0 of the IBAD is reserved for future compatibility. This bit will always read 0.{iic_regs}

3.3.2 IIC Frequency Divider Register

Register address: Base address + $0001

	7	6	5	4	3	2	1	0
R W	IBC7	IBC6	IBC5	IBC4	IBC3	IBC2	IBC1	IBC0
RESET:	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-2 IIC Bus Frequency Divider Register (IBFD)

Read and write anytime{iic_regs}

IBC7–IBC0 — I-Bus Clock Rate 7–0

This field is used to prescale the clock for bit rate selection. {iic_div} The bit clock generator is implemented as a prescale divider - IBC7-6, prescaled shift register - IBC5-3 select the prescaler divider and IBC2-0 select the shift register tap point. {iic_div}The IBC bits are decoded to give the Tap and Prescale values as shown in **Table 3-2** {iic_div}

Table 3-2 I-Bus Tap and Prescale Values

IBC2-0 (bin)	SCL Tap (clocks)	SDA Tap (clocks)
000	5	1
001	6	1
010	7	2
011	8	2
100	9	3
101	10	3
110	12	4
111	15	4

IBC5-3 (bin)	scl2start (clocks)	scl2stop (clocks)	scl2tap (clocks)	tap2tap (clocks)
000	2	7	4	1
001	2	7	4	2
010	2	9	6	4
011	6	9	6	8
100	14	17	14	16
101	30	33	30	32
110	62	65	62	64
111	126	129	126	128

Table 3-3 Multiplier Factor

IBC7-6	MUL
00	01
01	02
10	04
11	RESERVED

The number of clocks from the falling edge of SCL to the first tap (Tap[1]) is defined by the values shown in the scl2tap column of **Table 3-2**, all subsequent tap points are separated by 2^{IBC5-3} as shown in the tap2tap column in **Table 3-2**.{iic_div} The SCL Tap is used to generated the SCL period and the SDA Tap is used to determine the delay from the falling edge of SCL to SDA changing, the SDA hold time.{iic_div}

IBC7-6 defines the multiplier factor MUL. {iic_div, iic_ack_addon}The values of MUL are shown in the **Table 3-3**{iic_div, iic_ack_addon}

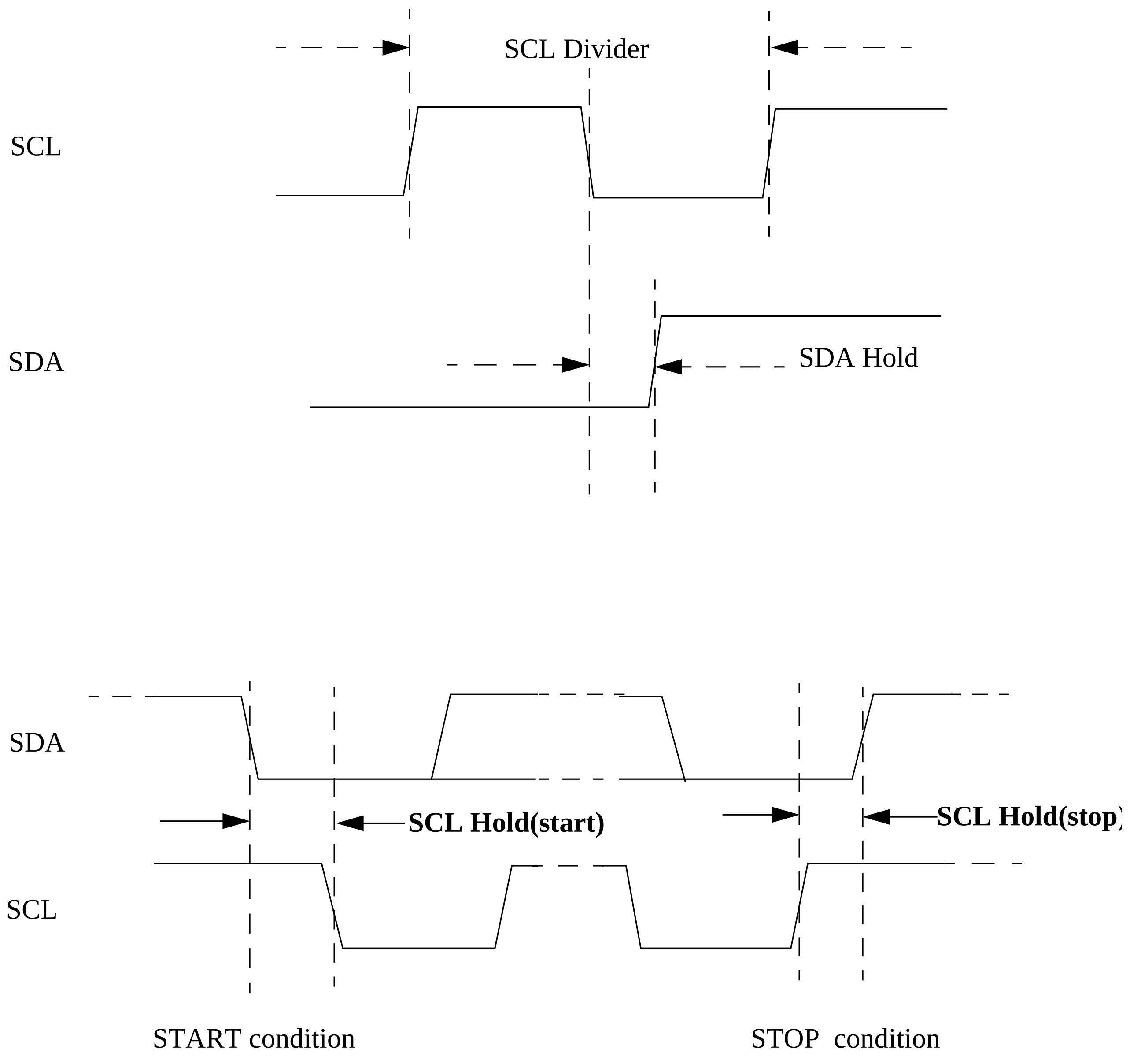


Figure 3-3 SCL divider and SDA hold

The equation used to generate the divider values from the IBFD bits is:

SCL Divider = MUL x {2 x (scl2tap + [(SCL_Tap -1) x tap2tap] + 2)}{iic_div}

The SDA hold delay is equal to the CPU clock period multiplied by the SDA Hold value shown in **Table 3-4**. {iic_div}The equation used to generate the SDA Hold value from the IBFD bits is:

SDA Hold = MUL x {scl2tap + [(SDA_Tap - 1) x tap2tap] + 3} {iic_div}

The equation for SCL Hold values to generate the start and stop conditions from the IBFD bits is:

SCL Hold(start) = MUL x [scl2start + (SCL_Tap - 1) x tap2tap] {iic_div}

SCL Hold(stop) = MUL x [scl2stop + (SCL_Tap - 1) x tap2tap] {iic_div}

Table 3-4 IIC Divider and Hold Values

IBC[7:0] (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL=1				
00	20	7	6	11
01	22	7	7	12
02	24	8	8	13
03	26	8	9	14
04	28	9	10	15
05	30	9	11	16
06	34	10	13	18
07	40	10	16	21
08	28	7	10	15
09	32	7	12	17
0A	36	9	14	19
0B	40	9	16	21
0C	44	11	18	23
0D	48	11	20	25
0E	56	13	24	29
0F	68	13	30	35
10	48	9	18	25
11	56	9	22	29
12	64	13	26	33
13	72	13	30	37
14	80	17	34	41
15	88	17	38	45
16	104	21	46	53
17	128	21	58	65

IBC[7:0] (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
18	80	9	38	41
19	96	9	46	49
1A	112	17	54	57
1B	128	17	62	65
1C	144	25	70	73
1D	160	25	78	81
1E	192	33	94	97
1F	240	33	118	121
20	160	17	78	81
21	192	17	94	97
22	224	33	110	113
23	256	33	126	129
24	288	49	142	145
25	320	49	158	161
26	384	65	190	193
27	480	65	238	241
28	320	33	158	161
29	384	33	190	193
2A	448	65	222	225
2B	512	65	254	257
2C	576	97	286	289
2D	640	97	318	321
2E	768	129	382	385
2F	960	129	478	481
30	640	65	318	321
31	768	65	382	385
32	896	129	446	449
33	1024	129	510	513
34	1152	193	574	577
35	1280	193	638	641
36	1536	257	766	769
37	1920	257	958	961

IBC[7:0] (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
38	1280	129	638	641
39	1536	129	766	769
3A	1792	257	894	897
3B	2048	257	1022	1025
3C	2304	385	1150	1153
3D	2560	385	1278	1281
3E	3072	513	1534	1537
3F	3840	513	1918	1921
MUL=2				
40	40	14	12	22
41	44	14	14	24
42	48	16	16	26
43	52	16	18	28
44	56	18	20	30
45	60	18	22	32
46	68	20	26	36
47	80	20	32	42
48	56	14	20	30
49	64	14	24	34
4A	72	18	28	38
4B	80	18	32	42
4C	88	22	36	46
4D	96	22	40	50
4E	112	26	48	58
4F	136	26	60	70
50	96	18	36	50
51	112	18	44	58
52	128	26	52	66
53	144	26	60	74
54	160	34	68	82
55	176	34	76	90
56	208	42	92	106

IBC[7:0] (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
57	256	42	116	130
58	160	18	76	82
59	192	18	92	98
5A	224	34	108	114
5B	256	34	124	130
5C	288	50	140	146
5D	320	50	156	162
5E	384	66	188	194
5F	480	66	236	242
60	320	34	156	162
61	384	34	188	194
62	448	66	220	226
63	512	66	252	258
64	576	98	284	290
65	640	98	316	322
66	768	130	380	386
67	960	130	476	482
68	640	66	316	322
69	768	66	380	386
6A	896	130	444	450
6B	1024	130	508	514
6C	1152	194	572	578
6D	1280	194	636	642
6E	1536	258	764	770
6F	1920	258	956	962
70	1280	130	636	642
71	1536	130	764	770
72	1792	258	892	898
73	2048	258	1020	1026
74	2304	386	1148	1154
75	2560	386	1276	1282
76	3072	514	1532	1538

IBC[7:0] (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
77	3840	514	1916	1922
78	2560	258	1276	1282
79	3072	258	1532	1538
7A	3584	514	1788	1794
7B	4096	514	2044	2050
7C	4608	770	2300	2306
7D	5120	770	2556	2562
7E	6144	1026	3068	3074
7F	7680	1026	3836	3842
MUL=4				
80	80	28	24	44
81	88	28	28	48
82	96	32	32	52
83	104	32	36	56
84	112	36	40	60
85	120	36	44	64
86	136	40	52	72
87	160	40	64	84
88	112	28	40	60
89	128	28	48	68
8A	144	36	56	76
8B	160	36	64	84
8C	176	44	72	92
8D	192	44	80	100
8E	224	52	96	116
8F	272	52	120	140
90	192	36	72	100
91	224	36	88	116
92	256	52	104	132
93	288	52	120	148
94	320	68	136	164
95	352	68	152	180

IBC[7:0] (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
96	416	84	184	212
97	512	84	232	260
98	320	36	152	164
99	384	36	184	196
9A	448	68	216	228
9B	512	68	248	260
9C	576	100	280	292
9D	640	100	312	324
9E	768	132	376	388
9F	960	132	472	484
A0	640	68	312	324
A1	768	68	376	388
A2	896	132	440	452
A3	1024	132	504	516
A4	1152	196	568	580
A5	1280	196	632	644
A6	1536	260	760	772
A7	1920	260	952	964
A8	1280	132	632	644
A9	1536	132	760	772
AA	1792	260	888	900
AB	2048	260	1016	1028
AC	2304	388	1144	1156
AD	2560	388	1272	1284
AE	3072	516	1528	1540
AF	3840	516	1912	1924
B0	2560	260	1272	1284
B1	3072	260	1528	1540
B2	3584	516	1784	1796
B3	4096	516	2040	2052
B4	4608	772	2296	2308
B5	5120	772	2552	2564

IBC[7:0] (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
B6	6144	1028	3064	3076
B7	7680	1028	3832	3844
B8	5120	516	2552	2564
B9	6144	516	3064	3076
BA	7168	1028	3576	3588
BB	8192	1028	4088	4100
BC	9216	1540	4600	4612
BD	10240	1540	5112	5124
BE	12288	2052	6136	6148
BF	15360	2052	7672	7684

3.3.3 IIC Control Register

Register address: Base address + $0002

	7	6	5	4	3	2	1	0
R	IBEN	IBIE	MS/$\overline{\text{SL}}$	Tx/$\overline{\text{Rx}}$	TXAK	0	0	IBSWAI
W						RSTA	▨	
RESET:	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-4 IIC-Bus Control Register (IBCR)

Read and write anytime{iic_regs}

IBEN — I-Bus Enable

This bit controls the software reset of the entire IIC Bus module.

0 = The module is reset and disabled.{iic_disable} This is the power-on reset situation. When low the interface is held in reset but registers can still be accessed{iic_disable}

1 = The IIC Bus module is enabled. {iic_div, iic_ack, iic_receive, iic_transmit}This bit must be set before any other IBCR bits have any effect{iic_disable}

If the IIC Bus module is enabled in the middle of a byte transfer the interface behaves as follows: slave mode ignores the current transfer on the bus and starts operating whenever a subsequent start condition is detected.Master mode will not be aware that the bus is busy, hence if a start cycle is initiated then the current bus cycle may become corrupt. This would ultimately result in either the current bus master or the IIC Bus module losing arbitration, after which bus operation would return to normal.

IBIE — I-Bus Interrupt Enable

0 = Interrupts from the IIC Bus module are disabled. {iic_int} Note that this does not clear any currently pending interrupt condition.{iic_int}

1 = Interrupts from the IIC Bus module are enabled. {iic_int}An IIC Bus interrupt occurs provided the IBIF bit in the status register is also set.{iic_int}

MS/$\overline{\text{SL}}$ — Master/Slave mode select bit

Upon reset, this bit is cleared. When this bit is changed from 0 to 1, a START signal is generated on the bus, and the master mode is selected.{iic_receive, iic_transmit} When this bit is changed from 1 to 0, a STOP signal is generated and the operation mode changes from master to slave.{iic_receive, iic_transmit} A STOP signal should only be generated if the IBIF flag is set. MS/$\overline{\text{SL}}$ is cleared without generating a STOP signal when the master loses arbitration.

0 = Slave Mode
1 = Master Mode

Tx/$\overline{\text{Rx}}$ — Transmit/Receive mode select bit

This bit selects the direction of master and slave transfers. {iic_receive, iic_transmit} When addressed as a slave this bit should be set by software according to the SRW bit in the status register. In master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high.

0 = Receive
1 = Transmit

TXAK — Transmit Acknowledge enable

This bit specifies the value driven onto SDA during data acknowledge cycles for both master and slave receivers.{iic_receive, iic_transmit} The IIC module will always acknowledge address matches, provided it is enabled, regardless of the value of TXAK. {iic_ack}Note that values written to this bit are only used when the IIC Bus is a receiver, not a transmitter.

0 = An acknowledge signal will be sent out to the bus at the 9th clock bit after receiving one byte data{iic_receive, iic_transmit}
1 = No acknowledge signal response is sent (i.e., acknowledge bit = 1){iic_receive, iic_transmit}

RSTA — Repeat Start

Writing a 1 to this bit will generate a repeated START condition on the bus, provided it is the current bus master. {iic_receive, iic_transmit}This bit will always be read as a low.{iic_regs, iic_receive, iic_transmit} Attempting a repeated start at the wrong time, if the bus is owned by another master, will result in loss of arbitration.

1 = Generate repeat start cycle

RESERVED

Bit 1 of the IBCR is reserved for future compatibility. This bit will always read 0.

{iic_regs}

IBSWAI — I-Bus Interface Stop in WAIT mode

0 = IIC Bus module clock operates normally{iic_wait}
1 = Halt IIC Bus module clock generation in WAIT mode{iic_wait}

Wait mode is entered via execution of a CPU WAI instruction. In the event that the IBSWAI bit is set, all clocks internal to the IIC will be stopped and any transmission currently in progress will halt.{iic_wait} If the CPU were woken up by a source other than the IIC module, then clocks would restart and the IIC would continue where it left off in the previous transmission.{iic_wait} It is not possible for the IIC to wake up the CPU when its internal clocks are stopped.

If it were the case that the IBSWAI bit was cleared when the WAI instruction was executed, the IIC internal clocks and interface would remain alive, continuing the operation which was currently underway. It is also possible to configure the IIC such that it will wake up the CPU via an interrupt at the conclusion of the current operation. See the discussion on the IBIF and IBIE bits in the IBSR and IBCR, respectively.

3.3.4 IIC Status Register

Register address: Base address + $0003

	7	6	5	4	3	2	1	0
R	TCF	IAAS	IBB	IBAL	0	SRW	IBIF	RXAK
W	▨	▨	▨	IBAL	▨	▨	IBIF	▨
RESET:	1	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-5 IIC Bus Status Register (IBSR)

This status register is read-only with exception of bit 1 (IBIF) and bit 4 (IBAL), which are software clearable{iic_regs}

TCF — Data transferring bit

While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. Note that this bit is only valid during or immediately following a transfer to the IIC module or from the IIC module.{iic_int}
0 = Transfer in progress
1 = Transfer complete

IAAS — Addressed as a slave bit

When its own specific address (I-Bus Address Register) is matched with the calling address, this bit is set.{iic_slave}The CPU is interrupted provided the IBIE is set.{iic_int}Then the CPU needs to check the SRW bit and set its $\text{Tx}/\overline{\text{Rx}}$ mode accordingly.Writing to the I-Bus Control Register clears this bit.{iic_int}
0 = Not addressed
1 = Addressed as a slave

IBB — Bus busy bit

This bit indicates the status of the bus. When a START signal is detected, the IBB is set. If a STOP signal is detected, IBB is cleared and {iic_receive, iic_transmit}
0 = the bus enters idle state.
1 = Bus is busy

IBAL — Arbitration Lost

The arbitration lost bit (IBAL) is set by hardware when the arbitration procedure is lost. Arbitration is lost in the following circumstances:

1. SDA sampled low when the master drives a high during an address or data transmit cycle.{iic_arb}
2. SDA sampled low when the master drives a high during the acknowledge bit of a data receive cycle.{iic_arb}
3. A start cycle is attempted when the bus is busy.{iic_arb}
4. A repeated start cycle is requested in slave mode.{iic_arb}
5. A stop condition is detected when the master did not request it.{iic_arb}

This bit must be cleared by software, by writing a one to it. A write of zero has no effect on this bit.

RESERVED

Bit 3 of IBSR is reserved for future use. A read operation on this bit will return 0.{iic_regs}

SRW — Slave Read/Write

When IAAS is set this bit indicates the value of the R/W command bit of the calling address sent from the master. {iic_receive, iic_transmit}

This bit is only valid when the I-Bus is in slave mode, a complete address transfer has occurred with an address match and no other transfers have been initiated.

Checking this bit, the CPU can select slave transmit/receive mode according to the command of the master.
0 = Slave receive, master writing to slave
1 = Slave transmit, master reading from slave

IBIF — I-Bus Interrupt

The IBIF bit is set when one of the following conditions occurs:

– arbitration lost (IBAL bit set)
– byte transfer complete (TCF bit set)
– addressed as slave (IAAS bit set)

It will cause a processor interrupt request if the IBIE bit is set. This bit must be cleared by software, writing a one to it. A write of zero has no effect on this bit.

RXAK — Received Acknowledge

The value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion of 8 bits data transmission on the bus.{iic_ack} If RXAK is high, it means no acknowledge signal is detected at the 9th clock.{iic_ack}
0 = Acknowledge received
1 = No acknowledge received

3.3.5 IIC Data I/O Register

Register address

	7	6	5	4	3	2	1	0
R W	D7	D6	D5	D4	D3	D2	D1	D0
RESET:	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 3-6 IIC Bus Data I/O Register (IBDR)

In master transmit mode, when data is written to the IBDR a data transfer is initiated. {iic_transmit}The most significant bit is sent first. In master receive mode, reading this register initiates next byte data receiving.{iic_receive} In slave mode, the same functions are available after an address match has occurred.{iic_receive, iic_transmit}Note that the Tx/Rx bit in the IBCR must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. {iic_receive, iic_transmit} For instance, if the IIC is configured for master transmit but a master receive is desired, then reading the IBDR will not initiate the receive.

Reading the IBDR will return the last byte received while the IIC is configured in either master receive or slave receive modes. {iic_receive}The IBDR does not reflect every byte that is transmitted on the IIC bus, nor can software verify that a byte has been written to the IBDR correctly by reading it back.

In master transmit mode, the first byte of data written to IBDR following assertion of MS/$\overline{\text{SL}}$ is used for the address transfer and should com.prise of the calling address (in position D7-D1) concatenated with the required R/$\overline{\text{W}}$ bit (in position D0).

Section 4 Functional Description

4.1 General

This section provides a complete functional description of the IIC.

4.2 I-Bus Protocol

The IIC Bus system uses a Serial Data line (SDA) and a Serial Clock Line (SCL) for data transfer. All devices connected to it must have open drain or open collector outputs. Logic AND function is exercised on both lines with external pull-up resistors. The value of these resistors is system dependent.

Normally, a standard communication is composed of four parts: START signal, slave address transmission, data transfer and STOP signal. They are described briefly in the following sections and illustrated in **Figure 4-1**.

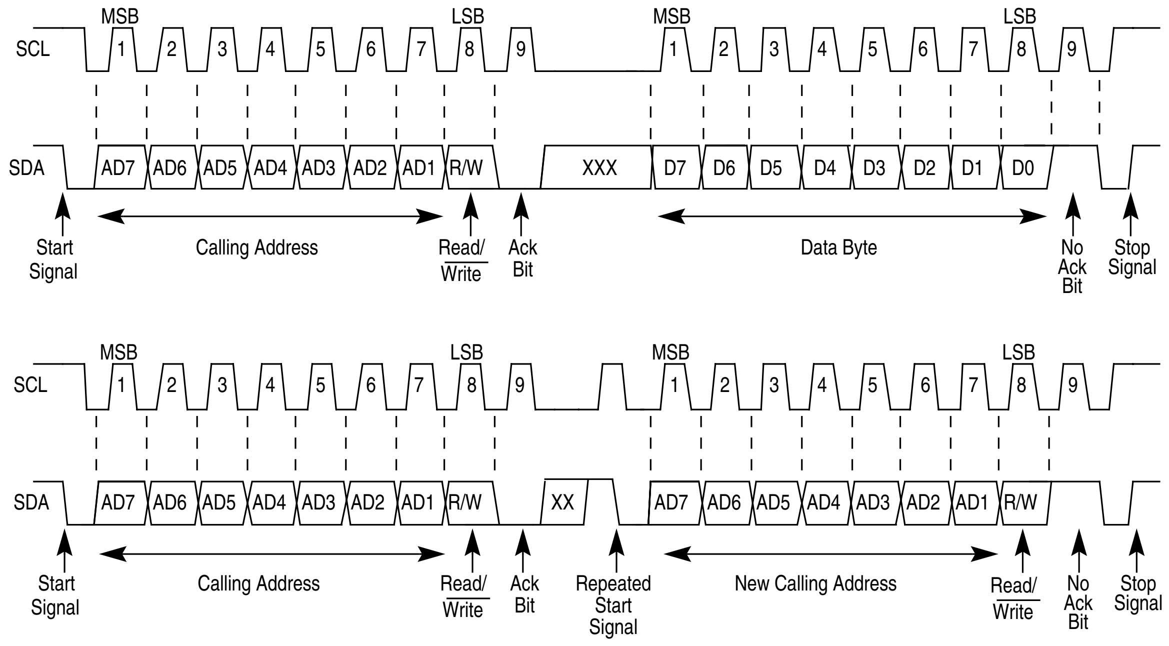


Figure 4-1 IIC-Bus Transmission Signals

4.2.1 START Signal

When the bus is free, i.e. no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START signal.As shown in Figure 4-1, a START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

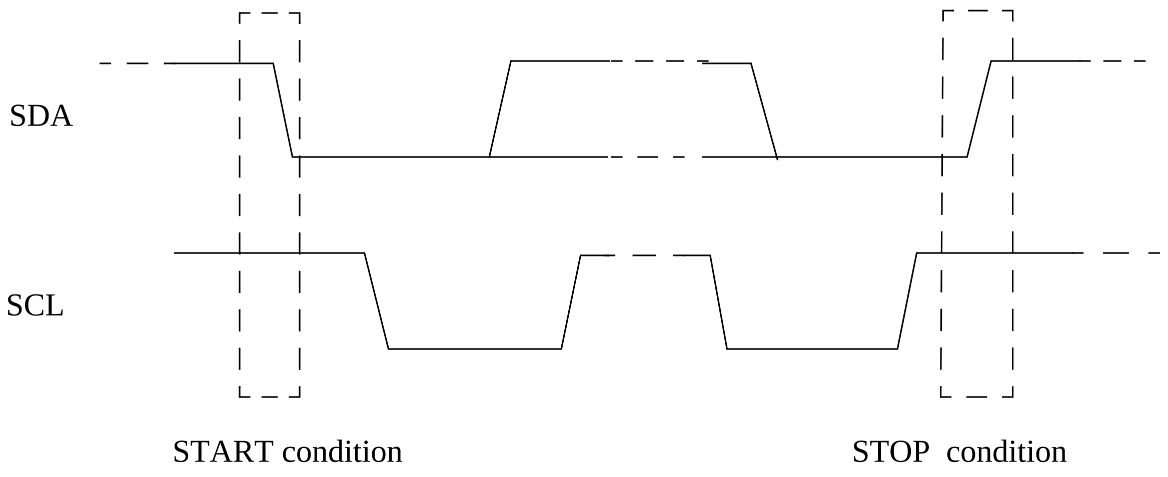


Figure 4-2 Start and Stop conditions

4.2.2 Slave Address Transmission

The first byte of data transfer immediately after the START signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/W bit. The R/W bit tells the slave the desired direction of data transfer.

1 = Read transfer, the slave transmits data to the master.
0 = Write transfer, the master transmits data to the slave.

Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see Figure 4-1).

No two slaves in the system may have the same address. If the IIC Bus is master, it must not transmit an address that is equal to its own slave address. The IIC Bus cannot be master and slave at the same time.However, if arbitration is lost during an address cycle the IIC Bus will revert to slave mode and operate correctly even if it is being addressed by another master.

4.2.3 Data Transfer

Once successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/W bit sent by the calling master

All transfers that come after an address cycle are referred to as data transfers, even if they carry sub-address information for the slave device.

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in Figure 4-1. There is one clock pulse on SCL for each data bit, the MSB being transferred first. Each data byte has to be followed by an acknowledge bit, which is signalled from the receiving device by pulling the SDA low at the ninth clock. So one complete data byte transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop signal to abort the data transfer or a start signal (repeated start) to commence a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it means 'end of data' to the slave, so the slave releases the SDA line for the master to generate STOP or START signal.

4.2.4 STOP Signal

The master can terminate the communication by generating a STOP signal to free the bus. However, the master may generate a START signal followed by a calling command without generating a STOP signal first. This is called repeated START. A STOP signal is defined as a low-to-high transition of SDA while SCL at logical “1” (see **Figure 4-1**).

The master can generate a STOP even if the slave has generated an acknowledge at which point the slave must release the bus.

4.2.5 Repeated START Signal

As shown in **Figure 4-1**, a repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication. This is used by the master to communicate with another slave or with the same slave in different mode (transmit/receive mode) without releasing the bus.

4.2.6 Arbitration Procedure

The Inter-IC bus is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure, a bus master loses arbitration if it transmits logic “1” while another master transmits logic “0”. The losing masters immediately switch over to slave receive mode and stop driving SDA output. In this case the transition from master to slave mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

4.2.7 Clock Synchronization

Since wire-AND logic is performed on SCL line, a high-to-low transition on SCL line affects all the devices connected on the bus. The devices start counting their low period and once a device's clock has gone low, it holds the SCL line low until the clock high state is reached.However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see **Figure 4-2**). When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. There is then no difference between the device clocks and the state of the SCL line and all the devices start counting their high periods.The first device to complete its high period pulls the SCL line low again.

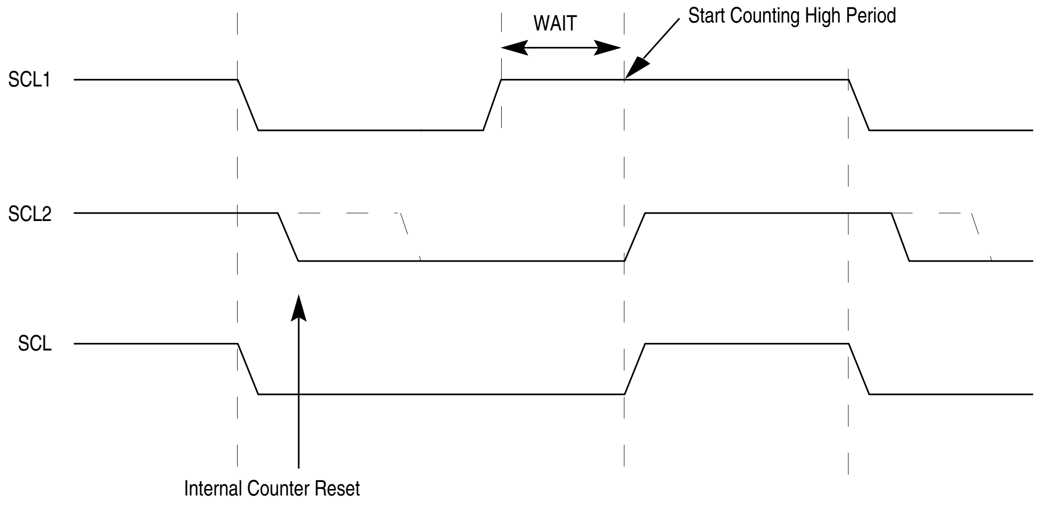


Figure 4-3 IIC-Bus Clock Synchronization

4.2.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such case, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

4.2.9 Clock Stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low the slave can drive SCL low for the required period and then release it.If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

4.3 Modes of Operation

The IIC functions the same in normal, special, and emulation modes. It has two low power modes, wait and stop modes

4.3.1 Run Mode

This is the basic mode of operation.

4.3.2 Wait Mode

IIC operation in wait mode can be configured. Depending on the state of internal bits, the IIC can operate normally when the CPU is in wait mode or the IIC clock generation can be turned off and the IIC module enters a power conservation state during wait mode. In the later case, any transmission or reception in progress stops at wait mode entry.

4.3.3 Stop Mode

The IIC is inactive in stop mode for reduced power consumption. The STOP instruction does not affect IIC register states.

Section 5 Initialization/Application Information

5.1 IIC Programming Examples

5.1.1 Initialization Sequence

Reset will put the IIC Bus Control Register to its default status. Before the interface can be used to transfer serial data, an initialization procedure must be carried out, as follows:

1. Update the Frequency Divider Register (IBFD) and select the required division ratio to obtain SCL frequency from system clock.
2. Update the IIC Bus Address Register (IBAD) to define its slave address.
3. Set the IBEN bit of the IIC Bus Control Register (IBCR) to enable the IIC interface system.
4. Modify the bits of the IIC Bus Control Register (IBCR) to select Master/Slave mode, Transmit/Receive mode and interrupt enable or not.

5.1.2 Generation of START

After completion of the initialization procedure, serial data can be transmitted by selecting the 'master transmitter' mode. If the device is connected to a multi-master bus system, the state of the IIC Bus Busy bit (IBB) must be tested to check whether the serial bus is free.

If the bus is free (IBB=0), the start condition and the first byte (the slave address) can be sent. The data written to the data register comprises the slave calling address and the LSB set to indicate the direction of transfer required from the slave.

The bus free time (i.e., the time between a STOP condition and the following START condition) is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period it may be necessary to wait until the IIC is busy after writing the calling address to the IBDR before proceeding with the following instructions. This is illustrated in the following example.

An example of a program which generates the START signal and transmits the first byte of data (slave address) is shown below:

CHFLAG	BRSET	IBSR,#$20,*	;WAIT FOR IBB FLAG TO CLEAR
TXSTART	BSET	IBCR,#$30	;SET TRANSMIT AND MASTER MODE;i.e. GENERATE START CONDITION
	MOVB	CALLING,IBDR	;TRANSMIT THE CALLING ADDRESS, D0=R/W
IBFREE	BRCLR	IBSR,#$20,*	;WAIT FOR IBB FLAG TO SET

5.1.3 Post-Transfer Software Response

Successful transmission or reception of a byte will set the TCF (data transferring) bit and the IBIF (interrupt flag) bit in the IBSR status register. An interrupt service routine can be called by this action if the IBIE (interrupt enable) bit in the IBCR control register is set. The IBIF (interrupt flag) bit can be cleared by writing 1 (in the interrupt service routine, if interrupts are used).

The TCF bit will be cleared to indicate data transfer in progress by reading the IBDR data register in receive mode or writing the IBDR in transmit mode. The TCF bit should not be used as a data transfer complete flag as the flag timing is dependent on a number of factors including the IIC bus frequency. This bit may not conclusively provide an indication of a transfer complete situation. It is recommended that transfer complete situations are detected using the IBIF flag

Software may service the IIC I/O in the main program by monitoring the IBIF bit if the interrupt function is disabled. Note that polling should monitor the IBIF bit rather than the TCF bit since their operation is different when arbitration is lost.

Note that when an interrupt occurs at the end of the address cycle the master will always be in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/W bit in IBDR, then the Tx/Rx bit should be toggled at this stage.

During slave mode address cycles (IAAS=1) the SRW bit in the status register is read to determine the direction of the subsequent transfer and the Tx/Rx bit is programmed accordingly. For slave mode data cycles (IAAS=0) the SRW bit is not valid, the Tx/Rx bit in the control register should be read to determine the direction of the current transfer.

The following is an example of a software response by a 'master transmitter' in the interrupt routine .

ISR	BCLR	IBSR,#$02	;CLEAR THE IBIF FLAG
	BRCLR	IBCR,#$20,SLAVE	;BRANCH IF IN SLAVE MODE
	BRCLR	IBCR,#$10,RECEIVE	;BRANCH IF IN RECEIVE MODE
	BRSET	IBSR,#$01,END	;IF NO ACK, END OF TRANSMISSION
TRANSMIT	MOVB	DATABUF,IBDR	;TRANSMIT NEXT BYTE OF DATA

5.1.4 Generation of STOP

A data transfer ends with a STOP signal generated by the 'master' device. A master transmitter can simply generate a STOP signal after all the data has been transmitted. The following is an example showing how a stop condition is generated by a master transmitter.

MASTX	TST	TXCNT	;GET VALUE FROM THE TRANSMITING COUNTER
	BEQ	END	;END IF NO MORE DATA
	BRSET	IBSR,#$01,END	;END IF NO ACK
	MOVB	DATABUF,IBDR	;TRANSMIT NEXT BYTE OF DATA
	DEC	TXCNT	;DECREASE THE TXCNT
	BRA	EMASTX	;EXIT
END	BCLR	IBCR,#$20	;GENERATE A STOP CONDITION
EMASTX	RTI		;RETURN FROM INTERRUPT

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the transmit acknowledge bit (TXAK) before reading the 2nd last byte of data. Before reading the last byte of data, a STOP signal must be generated first. The following is an example showing how a STOP signal is generated by a master receiver.

MASR	DEC	RXCNT	;DECREASE THE RXCNT
	BEQ	ENMASR	;LAST BYTE TO BE READ
	MOVB	RXCNT,D1	;CHECK SECOND LAST BYTE
	DEC	D1	;TO BE READ
	BNE	NXMAR	;NOT LAST OR SECOND LAST
LAMAR	BSET	IBCR,#$08	;SECOND LAST, DISABLE ACK
			;TRANSMITTING
	BRA	NXMAR	
ENMASR	BCLR	IBCR,#$20	;LAST ONE, GENERATE ‘STOP’ SIGNAL
NXMAR	MOVB	IBDR,RXBUF	;READ DATA AND STORE
	RTI		

5.1.5 Generation of Repeated START

At the end of data transfer, if the master still wants to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown.

RESTART	BSET	IBCR,#$04	;ANOTHER START (RESTART)
	MOVB	CALLING,IBDR	;TRANSMIT THE CALLING ADDRESS;D0=R/W

5.1.6 Slave Mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received . If IAAS is set, software should set the transmit/receive mode select bit (Tx/Rx bit of IBCR) according to the R/W command bit (SRW). Writing to the IBCR clears the IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred, interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR, for slave transmits, or dummy reading from IBDR, in slave receive mode. The slave will drive SCL low in-between byte transfers, SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means an 'end of data' signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a STOP signal.

5.1.7 Arbitration Lost

If several masters try to engage the bus simultaneously, only one master wins and the others lose arbitration. The devices which lost arbitration are immediately switched to slave receive mode by the hardware. Their data output to the SDA line is stopped, but SCL is still generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with IBAL=1 and MS/SL=0. If one master attempts to start transmission while the bus is being engaged by another master, the hardware will inhibit the transmission; switch the MS/SL bit from 1 to 0 without generating STOP condition; generate an interrupt to CPU and set the IBAL to indicate that the attempt to engage the bus is failed. When considering these cases, the slave service routine should test the IBAL first and the software should clear the IBAL bit if it is set.

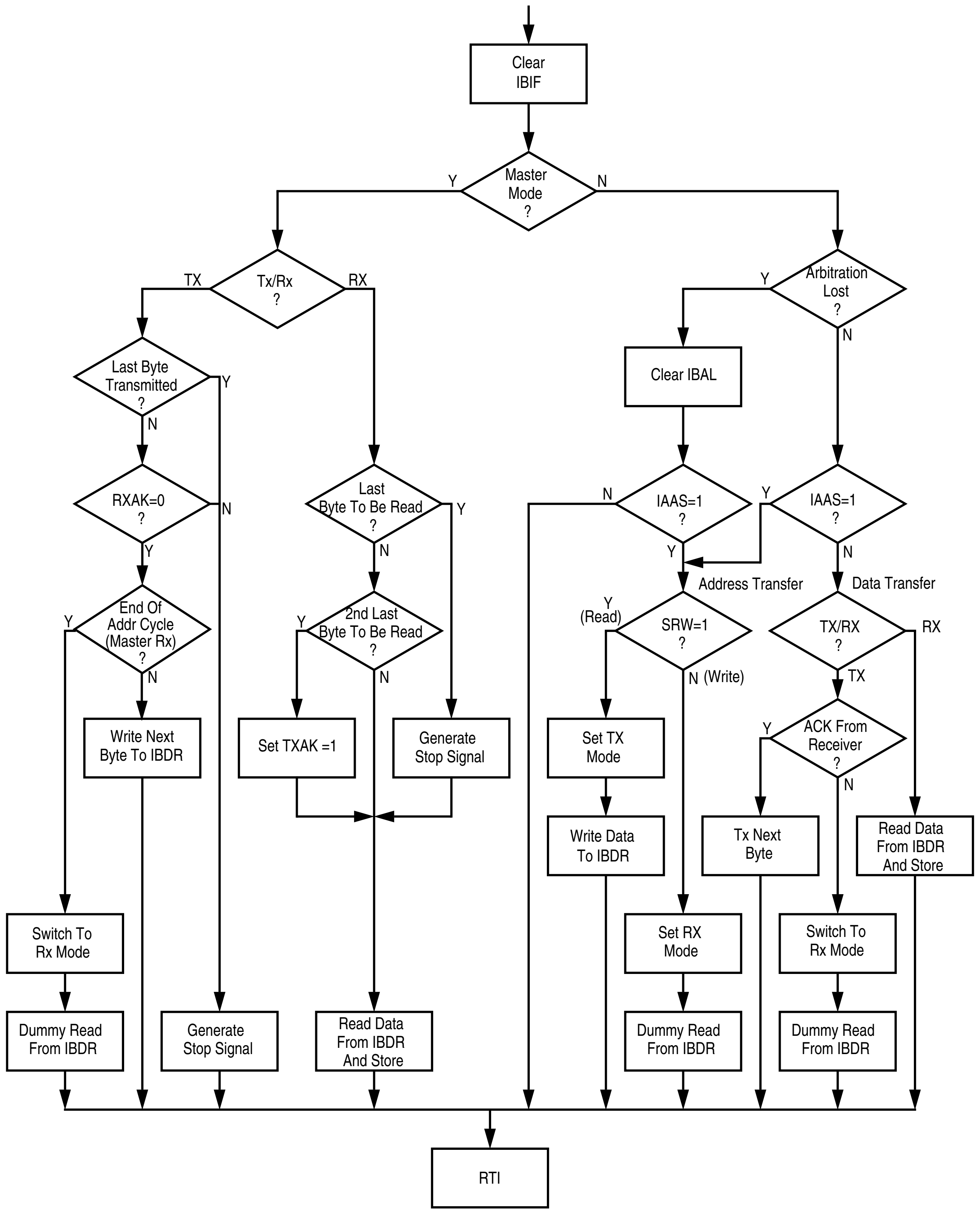


Figure 5-1 Flow-Chart of Typical IIC Interrupt Routine

Section 6 Resets

6.1 General

The reset state of each individual bit is listed within the Register Description section (see **Section 3 Memory Map/Register Definition**) which details the registers and their bit-fields.

Section 7 Interrupts

7.1 General

IIC uses only one interrupt vector.

Table 7-1 Interrupt Summary

Interrupt	Offset	Vector	Priority	Source	Description
IIC Interrupt	-	-	-	IBAL, TCF, IAAS bits in IBSR register	When either of IBAL, TCF or IAAS bits is set may cause an interrupt based on Arbitration lost, Transfer Complete or Address Detect conditions.

7.2 Interrupt Description

Internally there are three types of interrupts in IIC. The interrupt service routine can determine the interrupt type by reading the Status Register.

IIC Interrupt can be generated on

1. Arbitration Lost condition (IBAL bit set)
2. Byte Transfer condition (TCF bit set)
3. Address Detect condition (IAAS bit set)

The IIC interrupt is enabled by the IBIE bit in the IIC Control Register. It must be cleared by writing ‘1’ to the IBIF bit in the interrupt service routine.

Block Guide End Sheet

FINAL PAGE OF
46
PAGES