SECTION 15 I^2C MODULE

15.1 OVERVIEW

I^2C is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange between devices. This two-wire bus minimizes the interconnection between devices.

This bus is suitable for applications requiring occasional communications over a short distance between many devices. The flexible I^2C allows additional devices to be connected to the bus for expansion and system development.

The interface operates up to 100 kbps with maximum bus loading and timing.

The I^2C system is a true multimaster bus including collision detection and arbitration that prevents data corruption if two or more masters attempt to control the bus simultaneously. This feature allows for complex applications with multiprocessor control. It can also be used for rapid testing and alignment of end products via external connections to an assembly line computer.

15.2 INTERFACE FEATURES

The I^2C module has the following key features:

- Compatibility with I^2C Bus standard
- Multimaster operation
- Software-programmable for one of 64 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

Figure 15-1 is a block diagram of the I^2C module.

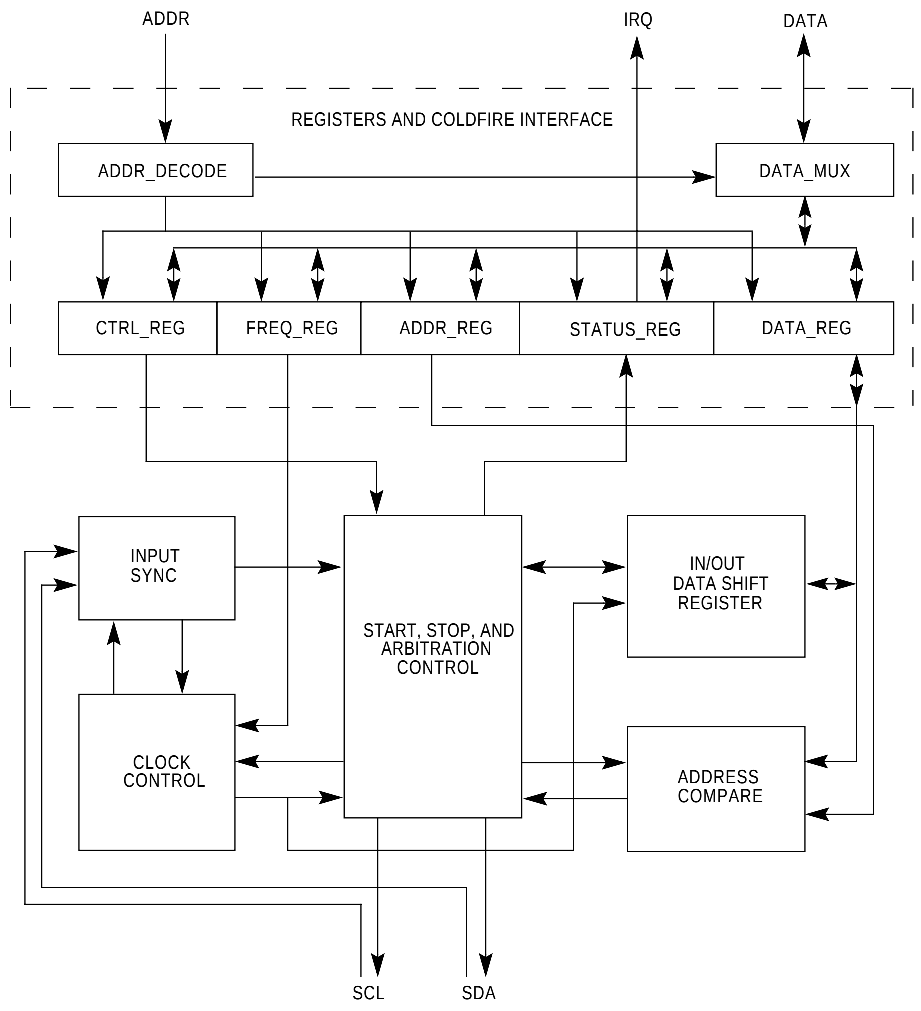

Figure 15-1. I^{2}C Module Block Diagram

15.3 I^{2}C SYSTEM CONFIGURATION

I^{2}C module uses a serial data line (SDA) and a serial clock line (SCL) for data transfer. All devices connected to these two signals must have open drain or open collector outputs. The logic AND function is exercised on both lines with external pullup resistors.

The default state of I^{2}C is as a slave receiver out of reset. Thus, when not programmed to be a master or responding to a slave transmit address, the I^{2}C module should always return to the default state of slave receiver (check 15.6.1 Initialization Sequence for exceptions).

NOTE

This I^{2}C module is designed to be compatible with the I^{2}C bus protocol from Philips. For further information on I^{2}C system configuration, protocol, and restrictions please refer to the Philips I^{2}C Standard

15.4 I^{2}C PROTOCOL

Normally, a standard communication is composed of four parts: (1) START signal, (2) slave address transmission, (3) data transfer, and (4) STOP signal. They are described briefly in the following sections and illustrated in Figure 15-2.

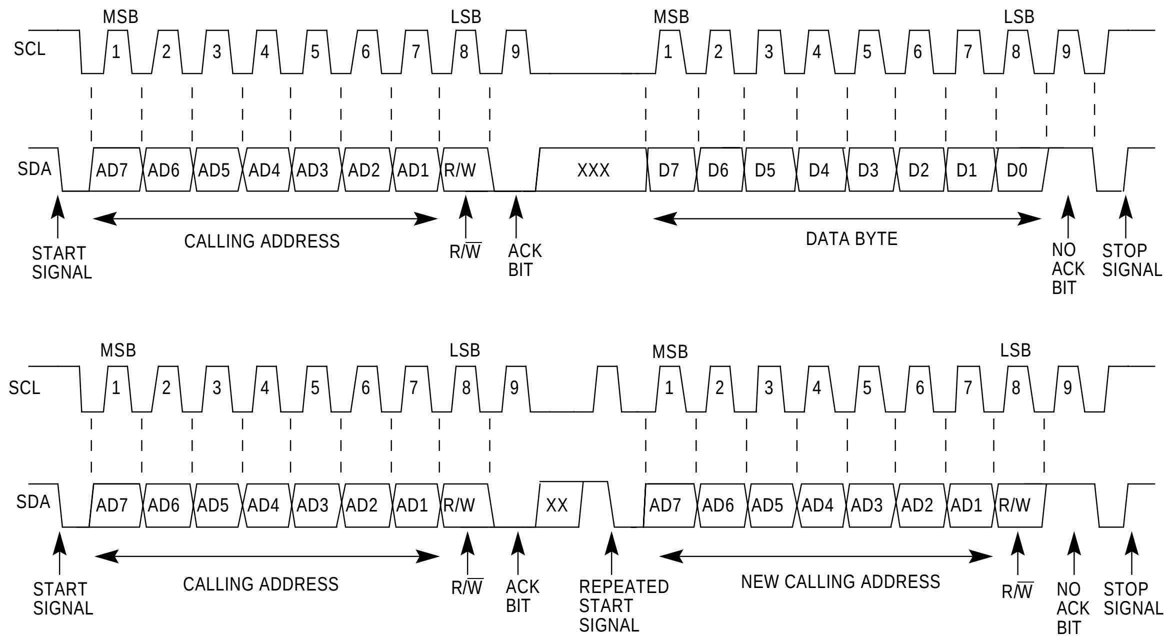

Figure 15-2. I^{2}C Standard Communication Protocol

15.4.1 START Signal

When the bus is free, i.e., no master device is engaging the bus (both SCL and SDA lines are at logic high), a master can initiate communication by sending a START signal. As shown in Figure 15-2, 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 can contain several bytes of data) and awakens all slaves.

15.4.2 Slave Address Transmission

The first byte of data transferred by the master immediately after the START signal is the slave address. This is a seven-bit calling address followed by a R/W bit. The R/W bit tells the slave data transfer direction. No two slaves in the system can have the same address. In addition, if the I^2C is master, it must not transmit an address that is equal to its slave address. The I^2C cannot be master and slave at the same time.

Only the slave with an address that matches the one transmitted by the master will respond by returning an acknowledge bit by pulling the SDA low at the 9th clock (see Figure 15-2).

15.4.3 Data Transfer

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

Each data byte is 8 bits long. Data can be changed only while SCL is low and must be held stable while SCL is high, as shown in Figure 15-2. There is one clock pulse on SCL for each data bit with the MSB being transferred first. Each byte of data must be followed by an acknowledge bit, which is signalled from the receiving device by pulling the SDA low at the ninth clock. 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 start a new calling sequence.

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

15.4.4 Repeated START Signal

As shown in Figure 15-2, a repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication. The master uses this method to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

15.4.5 STOP Signal

The master can terminate the communication by generating a STOP signal to free the bus. However, the master can 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 is at logical 1 (see Figure 15-2). Note that a master can generate a STOP even if the slave has made an acknowledgment at which point the slave must release the bus.

15.4.6 Arbitration Procedure

I^{2}C is a true multimaster bus that allows more than one master to be connected on it. If two or more masters try to simultaneously control the bus, 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 devices. A data arbitration procedure determines the relative priority of the contending masters. 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, hardware sets I2SR[IAL] to indicate loss of arbitration.

15.4.7 Clock Synchronization

Because 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 when the master drives the SCL line low. Once a device 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 15-3). 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.

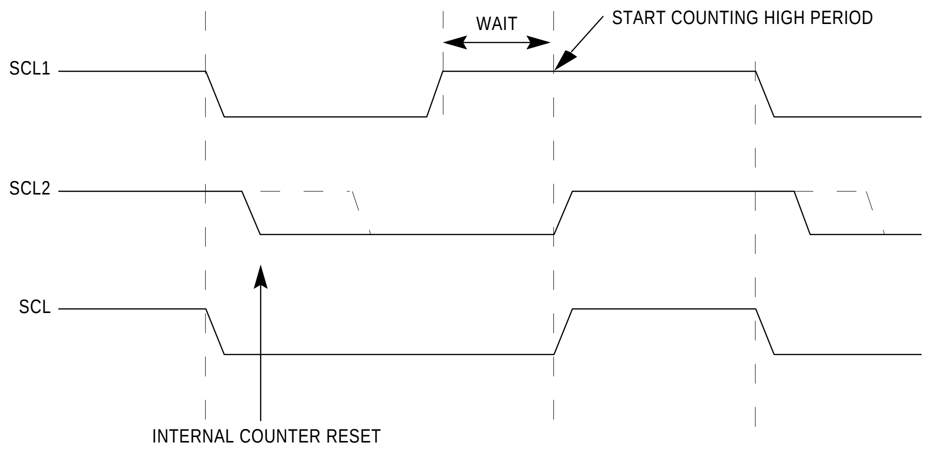

Figure 15-3. Synchronized Clock SCL

15.4.8 Handshaking

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

15.4.9 Clock Stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. 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, the resulting SCL bus signal low period is stretched.

15.5 PROGRAMMING MODEL

Internal configuration of the five registers used in the I^{2}C interface is discussed in the following paragraphs. Table 15-1 shows the programmer's model of the I^{2}C interface.

Table 15-1. I^{2}C Interface Programmer's Model

Address	I^{2}C Module Registers
MBAR+$280	I^{2}C Address Register (IADR)
MBAR+$284	I^{2}C Frequency Divider Register (IFDR)
MBAR+$288	I^{2}C Control Register (I2CR)
MBAR+$28C	I^{2}C Status Register (IBSR)
MBAR+$290	I^{2}C Data I/O Register (I2DR)

NOTE

External masters cannot access the MCF5307's on-chip memories or MBAR but can access any I^{2}C module register.

15.5.1 I^{2}C Address Register (IADR)

This register contains the address the I^{2}C will respond to when addressed as a slave; note that it is not the address sent on the bus during the address transfer.

I^{2}C Address Register (IADR) Address MBAR+$280

	7	6	5	4	3	2	1	0
	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	-
RESET	0	0	0	0	0	0	0	0

Read/Write Supervisor or User Mode

ADR7–ADR1 — Slave Address

Bit 1 to bit 7 contain the specific slave address to be used by the I^{2}C module.

Slave mode is the default I^{2}C mode for an address match on the bus.

15.5.2 I^{2}C Frequency Divider Register (IFDR)

I^{2}C Frequency Divider Reg. (IFDR) Address MBAR+$284

	7	6	5	4	3	2	1	0
	-	-	IC5	IC4	IC3	IC2	IC1	IC0
RESET	0	0	0	0	0	0	0	0

Read/Write Supervisor or User Mode

IC5–IC0 —I^{2}C Clock Rate 5–0

This field is used to prescale the clock for bit rate selection. Due to the potential slow rise and fall times of the SCL and SDA signals, bus signals are sampled at the prescaler frequency. The serial bit clock frequency is equal to the system clock divided by the divider shown in Table 15-2 (In previous implementations of the I^{2}C (e.g., MC68307), the IBC[5] (IC[5]) bit was not implemented. Clearing this bit in software maintains complete compatibility with such products.) Note that the IFDR frequency value can be changed at any point in a program.

Table 15-2. I^{2}C Prescaler Values

MBC5-0 (hex)	Divider (dec)	MBC5-0 (hex)	Divider (dec)
00	28	20	20
01	30	21	22
02	34	22	24
03	40	23	26
04	44	24	28
05	48	25	32
06	56	26	36
07	68	27	40
08	80	28	48
09	88	29	56
0A	104	2A	64
0B	128	2B	72
0C	144	2C	80
0D	160	2D	96
0E	192	2E	112
0F	240	2F	128
10	288	30	160
11	320	31	192
12	384	32	224
13	480	33	256
14	576	34	320
15	640	35	384
16	768	36	448
17	960	37	512
18	1152	38	640
19	1280	39	768
1A	1536	3A	896
1B	1920	3B	1024
1C	2304	3C	1280
1D	2560	3D	1536
1E	3072	3E	1792
1F	3840	3F	2048

15.5.3 I^{2}C Control Register (I2CR)

I^{2}C Control Register (I2CR) Address MBAR+$288

	7	6	5	4	3	2	1	0
	IEN	IIEN	MSTA	MTX	TXAK	RSTA	-	
RESET	0	0	0	0	0	0	0	0

Read/Write Supervisor or User Mode

IEN — I^{2}C Enable

This bit controls the software reset of the entire I^{2}C module.

1 = The I^{2}C module is enabled. This bit must be set before any other I2CR bits have any effect.
0 = The module is disabled, but registers can still be accessed.

If the I^{2}C module is enabled in the middle of a byte transfer, the interface behaves as follows: the 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; therefore, if a start cycle is initiated, the current bus cycle can become corrupt. This would ultimately result in either the current bus master or the I^{2}C module losing arbitration, after which bus operation would return to normal.

IIEN — I^{2}C Interrupt Enable

1 = Interrupts from the I^{2}C module are enabled. An I^{2}C interrupt occurs provided the IIF bit in the status register is also set.
0 = Interrupts from the I^{2}C module are disabled. This does not clear any currently pending interrupt condition.

MSTA — Master/Slave Mode Select Bit

At 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. When this bit is changed from 1 to 0, a STOP signal is generated and the operation mode changes from master to slave.

MSTA is cleared without generating a STOP signal when the master loses arbitration.

1 = Master Mode
0 = Slave Mode

MTX — Transmit/Receive mode select bit

This bit selects the direction of master and slave transfers. 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.

1 = Transmit
0 = Receive

TXAK — Transmit Acknowledge Enable

This bit specifies the value driven onto SDA during acknowledge cycles for both master and slave receivers. Note that writing this bit only applies when the I^{2}C bus is a receiver, not a transmitter.

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

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. This bit will always be read as a low. 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

15.5.4 I^{2}C Status Register (I2SR)

This status register is read-only with the exception of bit 1 (IIF) and bit 4 (IAL), which can be cleared by software. All bits are cleared on reset except bit 7 (ICF) and bit 0 (RXAK), which are set (=1) at reset.

I^{2}C Status Register (I2SR) Address MBAR+$28C

	7	6	5	4	3	2	1	0
	ICF	IAAS	IBB	IAL	-	SRW	IIF	RXAK
RESET	1	0	0	0	0	0	0	1

Read/Write Supervisor or User Mode

ICF — 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.

1 = Transfer complete
0 = Transfer in progress

IAAS — Addressed as a Slave Bit

When its own specific address (I^{2}C Address Register) is matched with the calling address, this bit is set. The CPU is interrupted provided the IIEN is set. Next, the CPU must check the SRW bit and set its TX/RX mode accordingly. Writing to the I^{2}C Control Register clears this bit.

1 = Addressed as a slave
0 = Not addressed

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, it is cleared.

1 =	Bus is busy
0 =	Bus is idle

IAL — Arbitration Lost

Hardware sets the arbitration lost bit (IAL) when the arbitration procedure is lost. Arbitration is lost in the following circumstances:

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

This bit must be cleared by software by writing a zero to it.

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. This bit is valid only when 1) a complete transfer has occurred and no other transfers have been initiated and 2) I^2C is a slave and has an address match. Checking this bit, the CPU can select slave transmit/receive mode according to the command of the master.

1 =	Slave transmit, master reading from slave
0 =	Slave receive, master writing to slave

IIF — I^2C Interrupt

The IIF bit is set when an interrupt is pending, which will cause a processor interrupt request (provided IIEN is set). IIF is set when one of the following events occurs:

1. Complete one byte transfer (set at the falling edge of the 9th clock)
2. Receive a calling address that matches its own specific address in slave-receive mode
3. Arbitration lost

This bit must be cleared by software by writing a zero to it in the interrupt routine.

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. If RXAK is high, it means no acknowledge signal has been detected at the 9th clock.

1 = No acknowledge received
0 = Acknowledge received

15.5.5 I^{2}C Data I/O Register (I2DR)

I^{2}C Data I/O Register (I2DR) Address MBAR+$290

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

Read/Write Supervisor or User Mode

When an address and R/W bit is written to the I2DR and the I^{2}C is the master, a transmission will start. When data is written to the I2DR, a data transfer is initiated. The most significant bit is sent first in both cases. In the master-receive mode, reading the I2DR register allows the read to occur but also initiates next byte data receiving. In slave mode, the same function is available after it is addressed.

15.6 I^{2}C PROGRAMMING EXAMPLES

15.6.1 Initialization Sequence

Reset will put the I^{2}C Control Register to its default status. Before the interface can transfer serial data, you must perform an initialization procedure as follows:

1. Update the Frequency Divider Register (IFDR) and select the required division ratio to obtain SCL frequency from the system bus clock.
2. Update the I^{2}C Address Register (IADR) to define its slave address.
3. Set the IEN bit of the I^{2}C Control Register (I2CR) to enable the I^{2}C bus interface system.
4. Modify the I2CR to select master/slave mode, transmit/receive mode, and interrupt-enable or not.

NOTE

During the initialization of the I^{2}C bus module, the user should check the IBB bit of the IBSR register. If the IBB bit is set when the I^{2}C module is enabled, then the following code sequence should be executed before proceeding with your normal initialization code. This issues a STOP command to the slave device, which will then places it into the idle state as if it were just power cycled on:

```
I2CR = $0
I2CR = $A
dummy read of I2DR
IBSR = $0
I2CR = $0
```

15.6.2 Generation of START

After completion of the initialization procedure, you can transmit serial data by selecting the “master transmitter'' mode. If the device is connected to a multimaster bus system, you must test the state of the I^{2}C Busy Bit (IBB) 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 address of the desired slave and the LSB is set to indicate the direction of transfer required.

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, you may have to wait until the I^{2}C is busy after writing the calling address to the I2DR before proceeding with the following instructions.

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

```
CHFLAG MOVE.B I2SR,-(A7)        ;Check the MBB bit of the I2SR
       BTST.B #5, (A7)+
       BNE.S  CHFLAG            ;If it is set, wait until it is clear

        TXSTARTMOVE.BI2CR,-(A7)       ;Set transmit mode
       BSET.B #4,(A7)
       MOVE.B (A7)+, I2CR
       MOVE.B I2CR, -(A7)       ;Set master mode
       BSET.B #5, (A7)          ;Generate START condition

       MOVE.B (A7)+, I2CR       ;
       MOVE.B CALLING,-(A7)     ;Transmit the calling address, D0=R/W

       MOVE.B (A7)+, I2DR       ;
IFREE  MOVE.B I2SR,-(A7)        ;Check the IBB bit of the I2SR.
                                ;If it is clear, wait until it is set.

       BTST.B #5, (A7)+         ;
       BEQ.S  IBFREE            ;
```

15.6.3 Post-Transfer Software Response

Transmission or reception of a byte will set the data transferring bit (ICF) to 1, which indicates one byte communication is finished. The interrupt bit (IIF) is also set. An interrupt will be generated if the interrupt function is enabled during initialization by setting the IIEN bit. Software must clear the IIF bit in the interrupt routine first. The ICF bit will be cleared by reading from the I^{2}C Data I/O Register (I2DR) in receive mode or writing to I2DR in transmit mode.

Software can service the I^{2}C I/O in the main program by monitoring the IIF bit if the interrupt function is disabled. Polling should monitor the IIF bit rather than the ICF bit because that operation is different when arbitration is lost.

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 I2DR[R/W], then the MTX bit should be toggled.

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 MTX bit is programmed accordingly. For slave-mode data cycles (IAAS=0), the SRW bit is not valid. The MTX 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 (see Figure 15-4).

```
I2SR   LEA.L  I2SR,-(A7)        ;Load effective address

       BCLR.B #1,(A7)+          ;Clear the IIF flag
           MOVE.B   I2CR,-(A7)        ;Push the address on stack,

       BTST.B #5,(A7)+          ;check the MSTA flag

       BEQ.S  SLAVE             ;Branch if slave mode
              MOVE.B      I2CR,-(A7)        ;Push the address on stack

       BTST.B #4,(A7)+          ;check the mode flag

       BEQ.S  RECEIVE           ;Branch if in receive mode
           MOVE.B   I2SR,-(A7)        ;Push the address on stack,

       BTST.B #0,(A7)+          ;check ACK from receiver

       BNE.B  END               ;If no ACK, end of transmission

        TRANSMITMOVE.BDATABUF,-(A7)   ; Stack data byte
        MOVE.B (A7)+, I2DR       ;Transmit next byte of data
```

15.6.4 Generation of STOP

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

```
MASTX  MOVE.B I2SR, -(A7)       ; If no ACK, branch to end
       BTST.B #0,(A7)+
       BNE.B  END
       MOVE.B TXCNT,D0          ;Get value from the transmitting counter

       BEQ.S  END               ;If no more data, branch to end

       MOVE.B DATABUF,-(A7)     ;Transmit next byte of data
       MOVE.B (A7)+,I2DR
       MOVE.B TXCNT,D0          ;Decrease the TXCNT

       SUBQ.L #1,D0
       MOVE.B D0,TXCNT
       BRA.S  EMASTX            ;
           Exit
END    LEA.L  I2CR,-(A7)        ;Generate a STOP condition
       BCLR.B #5,(A7)+
EMASTX RTE                      ; 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 next-to-last byte of data. Before reading the last byte of data, a STOP signal must first be generated. The following code is an example showing how a master receiver generates a STOP signal.

```
MASR   MOVE.B RXCNT,D0          ;Decrease RXCNT
       SUBQ.L #1,D0
       MOVE.B D0,RXCNT
       BEQ.S  ENMASR            ;Last byte to be read
       MOVE.B RXCNT,D1          ;Check second-to-last byte to be read

       EXTB.L D1
       SUBI.L #1,D1             ;
       BNE.S  NXMAR             ; Not last one or second last

LAMAR  BSET.B #3,I2CR           ;Disable ACK
       BRA    NXMAR

ENMASR BCLR.B #5,I2CR           ; Last one, generate 'STOP'signal
NXMAR  MOVE.B I2DR,RXBUF        ; Read data and store RTE
```

15.6.5 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 follows.

```
RESTARTMOVE.B I2CR,-(A7)        ; Another START (RESTART)
       BSET.B #2, (A7)
       MOVE.B (A7)+, I2CR
       MOVE.B CALLING,-(A7)     ;Transmit the calling address, D0=R/W-
       MOVE.B CALLING,-(A7)     ;
       MOVE.B (A7)+, I2DR
```

15.6.6 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 (MTX bit of I2CR) according to the R/W command bit (SRW). Writing to the I2CR clears the IAAS automatically. 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 can now be initiated by writing information to I2DR, for slave transmits, or read from I2DR, in slave-receive mode. A dummy read of the I2DR in slave/receive mode will release SCL, allowing the master to transmit data.

In the 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 read from I2DR then releases the SCL line so that the master can generate a STOP signal.

15.6.7 Arbitration Lost

If several devices try to engage the bus at the same time, only one becomes master and the others lose arbitration. The devices that 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 IAL=1 and MSTA=0. If one master tries to transmit or do a START while the bus is being engaged by another master, the hardware will: (1) inhibit the transmission, (2) switch the MSTA bit from 1 to 0 without generating STOP condition, (3) generate an interrupt to CPU and, (4) set the IAL to indicate the failed attempt to engage the bus. When considering these cases, the slave service routine should test the IAL first and the software should clear theIAL bit if it is set.

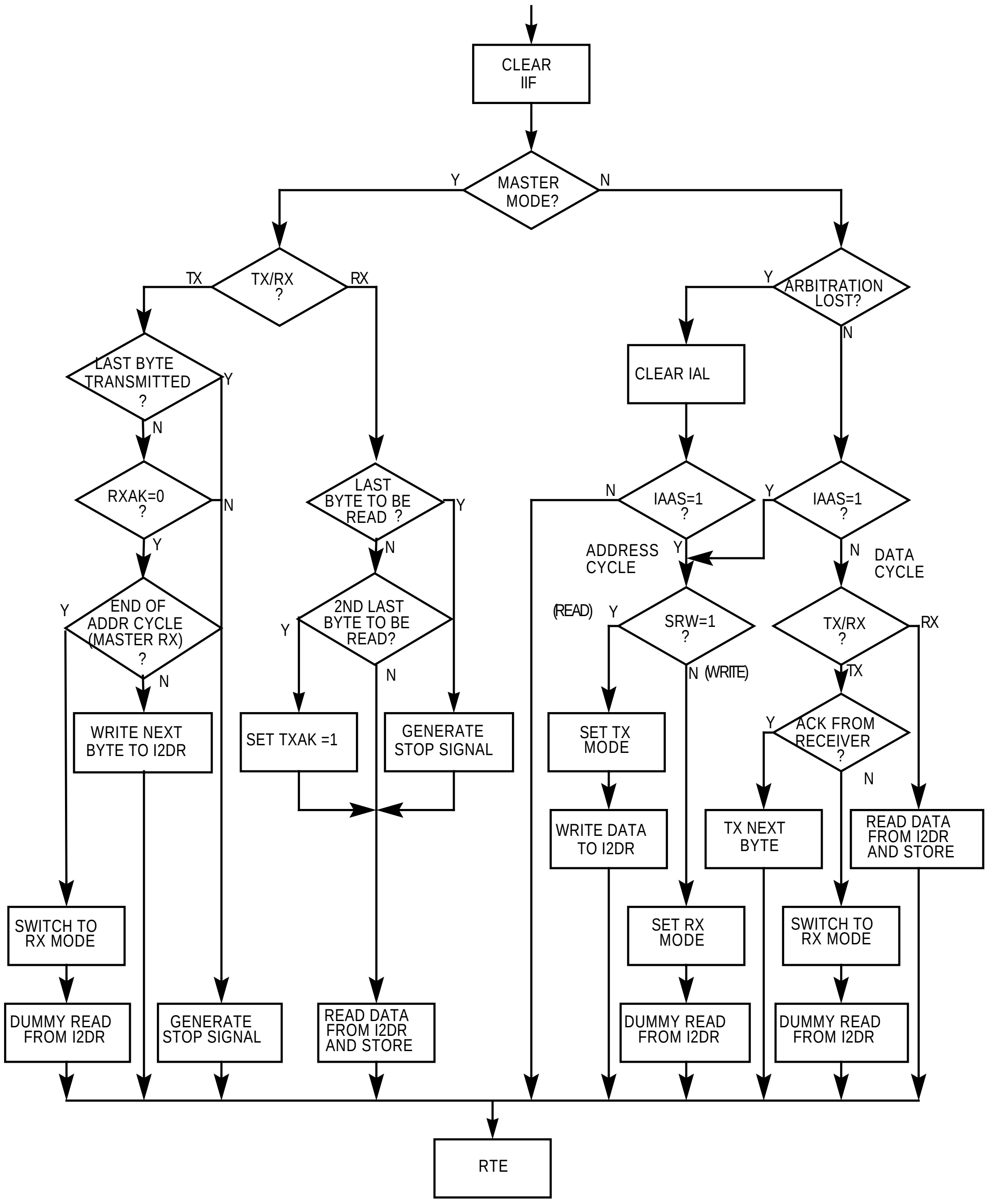

Figure 15-4. Flow-Chart of Typical I^{2}C Interrupt Routine