# SECTION 12
# M-BUS MODULE

## 12.1 OVERVIEW

Motorola bus (M-Bus) is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange between devices. It is compatible with the widely used $I^2C$ bus standard[1]. 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 M-Bus 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 M-Bus 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.

## 12.2 INTERFACE FEATURES

The M-Bus 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

---

1. $I^2C$-Bus is a proprietary Philips interface bus.

A block diagram of the complete M-Bus Module is shown in Figure 12-1.

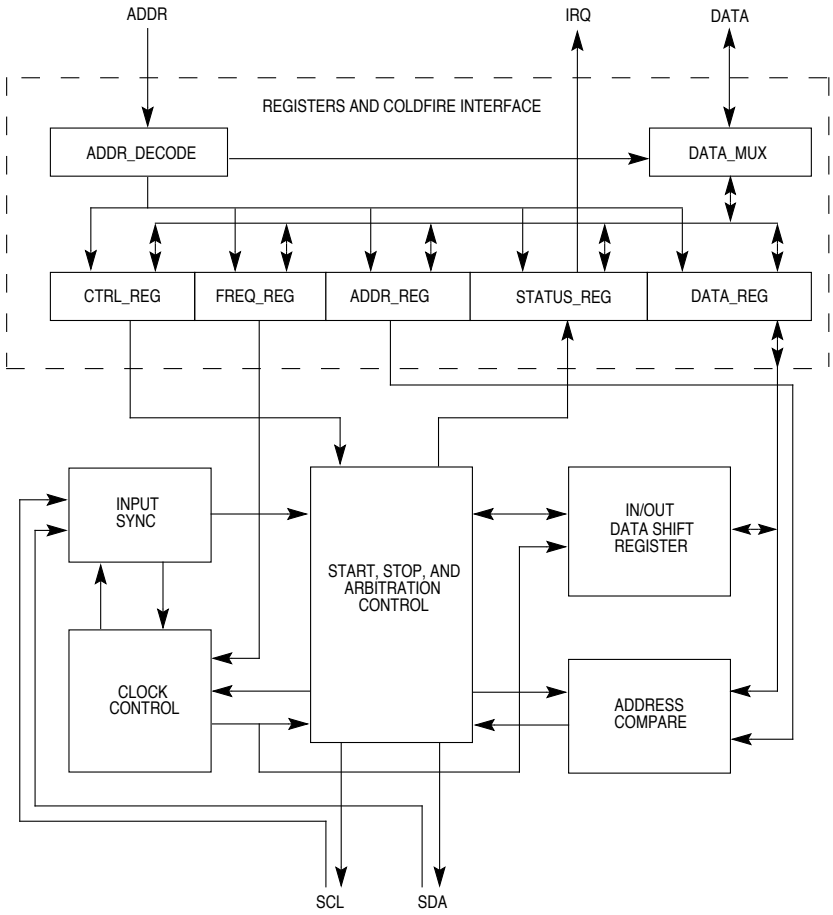


**Figure 12-1. M-Bus Module Block Diagram**

## 12.3 M-BUS SYSTEM CONFIGURATION

The M-Bus system 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 pullup resistors.

The default state of the M-Bus is as a slave reciever out of reset. Thus, when not programmed to be a master or responding to a slave transmit address, the M-Bus should always return to the default state of slave receiver.

**NOTE**

For further information on M-Bus system configuration, protocol, and restrictions please refer to the Philip's $I^2C$ standard

## 12.4 M-BUS 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 12-2.

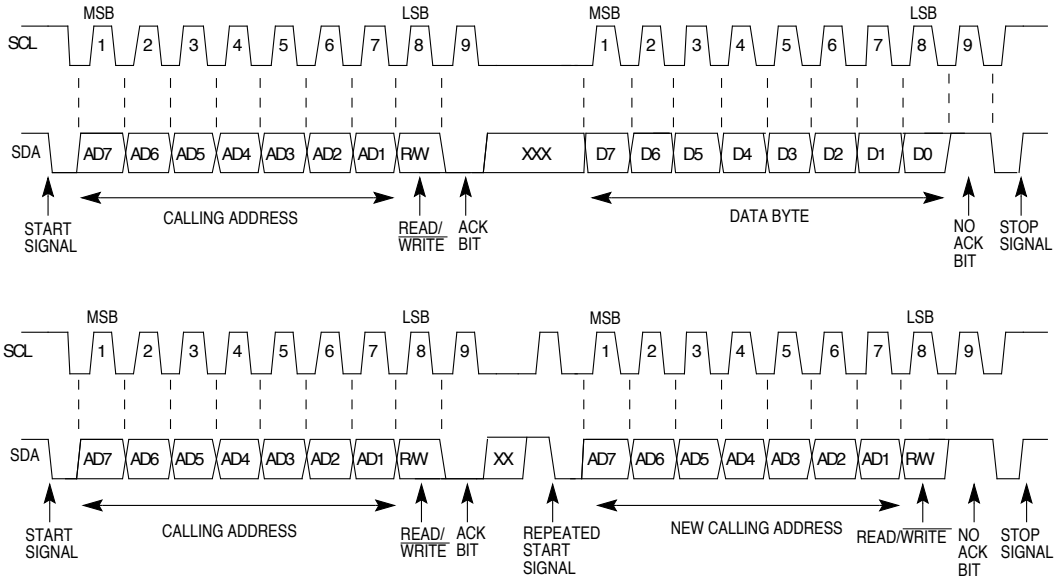


**Figure 12-2. M-Bus Standard Communication Protocol**

### 12.4.1 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 12-2, 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.

### 12.4.2 Slave Address Transmission

The first byte of data transfered by the master immediately after the START signal is the slave address. This is a seven-bit calling address followed by a R/$\overline{W}$ bit. The R/$\overline{W}$ bit tells the slave data transfer direction. No two slaves in the system can have the same address.

In addition, if the M-Bus is master, it must not transmit an address that is equal to its slave address. The M-Bus cannot be master and slave at the same time.

Only the slave with a calling 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 12-2).

### 12.4.3 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 12-2. There is one clock pulse on SCL for each data bit with the MSB being transferred first. Each byte 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 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. The slave releases the SDA line for the master to generate STOP or START signal.

### 12.4.4 Repeated START Signal

As shown in Figure 12-2, 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 different mode (transmit/receive mode) without releasing the bus.

### 12.4.5 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 at logical “1” (see Figure 12-2). Note that a master can generate a STOP even if the slave has done an acknowledgement at which point the slave must release the bus.

### 12.4.6 Arbitration Procedure

M-Bus 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 masters. 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 a status bit to indicate loss of arbitration.

### 12.4.7 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 12-3). 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.

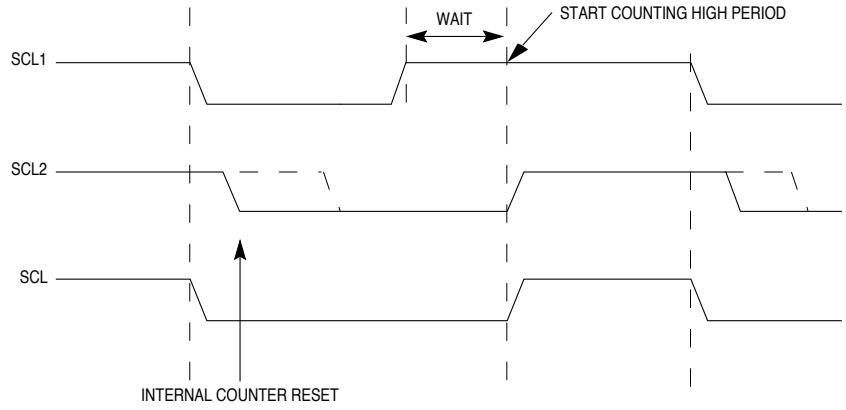


**Figure 12-3. Synchronized Clock SCL**

### 12.4.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices can hold the SCL 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.

### 12.4.9 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.

## 12.5 PROGRAMMING MODEL

Five registers are used in the M-Bus interface and the internal configuration of these registers is discussed in the following paragraphs.The programmer's model of the M-Bus interface is shown below in Table 12-1.

**Table 12-1. M-Bus Interface Programmer's Model**

| ADDRESS | M-BUS MODULE REGISTERS |
|---|---|
| MBAR+$1E0 | M-Bus Address register (MADR) |
| MBAR+$1E4 | M-Bus Frequency Divider Register (MFDR) |
| MBAR+$1E8 | M-Bus Control Register (MBCR) |
| MBAR+$1EC | M-Bus Status Register (MBSR) |
| MBAR+$1F0 | M-Bus Data I/O Register (MBDR) |

A block diagram of the M-Bus system is shown in Figure 12-1.

### 12.5.1 M-Bus Address Register (MADR)

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

M-Bus Address Register (MADR) — Address MBAR+$1E0

| | 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

Bits1 to 7 contain the specific slave address to be used by the M-Bus module.

**NOTE**

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

### 12.5.2 M-Bus Frequency Divider Register (MFDR)

M-Bus Frequency Divider Register (MFDR) — Address MBAR+$1E4

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | - | - | MBC5 | MBC4 | MBC3 | MBC2 | MBC1 | MBC0 |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Read/Write — Supervisor or User Mode

MBC5–MBC0 — M-Bus 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, the bus signals are sampled at the prescaler

frequency. The serial bit clock frequency is equal to the CPU clock divided by the divider shown in Table 12-2, which also shows the serial bit clock frequency for a 33MHz internal operating frequency[2]. Note that the MFDR frequency value can be changed at any point in a program.

**Table 12-2. MBUS Prescalar 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 |

2. In previous implementations of the M-Bus (e.g., MC68307), the MBC[5] bit was not implemented. Clearing this bit in software maintains complete compatibility with such products.

## 12.5.3 M-Bus Control Register (MBCR)

M-Bus Control Register (MBCR) — Address MBAR+$1E8

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | MEN | MIEN | MSTA | MTX | TXAK | RSTA | - | |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Read/Write | | | | | | Supervisor or User Mode | |

MEN — M-Bus Enable

This bit controls the software reset of the entire M-Bus module.

1 = The M-Bus module is enabled. This bit must be set before any other MBCR bits have any effect.
0 = The module is reset and disabled. This is the power-on reset situation. When low, the interface is held in reset but registers can still be accessed.

If the M-Bus 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 M-Bus module losing arbitration, after which bus operation would return to normal.

MIEN — M-Bus Interrupt Enable

1 = Interrupts from the M-Bus module are enabled. An M-Bus interrupt occurs provided the MIF bit in the status register is also set.
0 = Interrupts from the M-Bus 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 M-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. Note that this bit is not readable.

1 = Generate repeat start cycle

## 12.5.4 M-Bus Status Register (MBSR)

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

M-Bus Status Register (MBSR) — Address MBAR+$1EC

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | MCF | MAAS | MBB | MAL | - | SRW | MIF | RXAK |
| RESET | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | Read/Write | | | | | Supervisor or User Mode | | |

MCF — 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

MAAS — Addressed as a Slave Bit

When its own specific address (M-Bus Address Register) is matched with the calling address, this bit is set. The CPU is interrupted provided the MIEN is set. Next, the CPU must check the SRW bit and set its TX/RX mode accordingly.

Writing to the M-Bus Control Register clears this bit.

1 = Addressed as a slave
0 = Not addressed

MBB — Bus Busy Bit

This bit indicates the status of the bus. When a START signal is detected, the MBB is set. If a STOP signal is detected, it is cleared.

1 = Bus is busy
0 = Bus is idle

MAL — Arbitration Lost

Hardware sets the arbitration lost bit (MAL) 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 low to it.

SRW — Slave Read/Write

When MAAS 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) M-Bus 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

MIF — M-Bus Interrupt

The MIF bit is set when an interrupt is pending, which will cause a processor interrupt request (provided MIEN is set). MIF 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 low 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

### 12.5.5 M-Bus Data I/O Register (MBDR)

M-Bus Data I/O Register (MBDR) Address MBAR+$1F0

| | 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 MBDR and the M-Bus is the master, a transmission will start. When data is written to the MBDR, a data transfer is initiated. The most significant bit is sent first in both cases. In the master receive mode, reading the MBDR 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.

## 12.6 M-BUS PROGRAMMING EXAMPLES

### 12.6.1 Initialization Sequence

Reset will put the M-bus 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 (MFDR) and select the required division ratio to obtain SCL frequency from system clock.
2. Update the M-Bus Address Register (MADR) to define its slave address.
3. Set the MEN bit of the M-Bus Control Register (MBCR) to enable the M-Bus interface system.
4. Modify the bits of the M-Bus Control Register (MBCR) to select master/slave mode, transmit/receive mode, and interrupt-enable or not.

### 12.6.2 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 multi-master bus system, you must test the state of the M-Bus Busy Bit (MBB) to check whether the serial bus is free.

If the bus is free (MBB=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, you may have to wait until the

M-Bus is busy after writing the calling address to the MBDR 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:

```
CHFLAGMOVE.BMBSR,-(A7);       CHECK THE MBB BIT OF THE STATUS REGISTER. IF IT IS
BTST.B#5, (A7)+               SET, WAIT UNTIL IT IS CLEAR
BNE.SCHFLAG;

TXSTARTMOVE.BMBCR,-(A7);      SET TRANSMIT MODE
BSET.B#4,(A7)

MOVE.B(A7)+, MBCR
MOVE.BMBCR, -(A7);            SET MASTER MODE
BSET.B#5, (A7);               i.e. GENERATE START CONDITION
MOVE.B(A7)+, MBCR;
MOVE.BCALLING,-(A7);          TRANSMIT THE CALLING ADDRESS, D0=R/W
MOVE.B(A7)+, MBDR;
MBFREEMOVE.BMBSR,-(A7);       CHECK THE MBB BIT OF THE STATUS REGISTER. IF IT IS
BTST.B#5, (A7)+;              CLEAR, WAIT UNTIL IT IS SET
BEQ.SMBFREE;
```

### 12.6.3 Post-Transfer Software Response

Transmission or reception of a byte will set the data transferring bit (MCF) to 1, which indicates one byte communication is finished. The M-Bus interrupt bit (MIF) is also set ; an interrupt will be generated if the interrupt function is enabled during initialization by setting the MIEN bit. Software must clear the MIF bit in the interrupt routine first. The MCF bit will be cleared by reading from the M-Bus Data I/O Register (MDR) in receive mode or writing to MDR in transmit mode.

Software can service the M-bus I/O in the main program by monitoring the MIF bit if the interrupt function is disabled. Polling should monitor the MIF bit rather than the MCF 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 R/$\overline{W}$ bit in MBDR, then the MTX bit should be toggled at this stage.

During slave-mode address cycles (MAAS=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 (MAAS=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 12-4).

```
ISRLEA.LMBSR,-(A7);          LOAD EFFECTIVE ADDRESS
BCLR.B#1,(A7)+;              CLEAR THE MIF FLAG
MOVE.BMBCR,-(A7);            PUSH ADDRESS ON STACK,
BTST.B#5,(A7)+;              CHECK THE MSTA FLAG
BEQ.SSLAVE;                  BRANCH IF SLAVE MODE
MOVE.BMBCR,-(A7);            PUSH ADDRESS ON STACK
BTST.B#4,(A7)+;              CHECK THE MODE FLAG
BEQ.SRECEIVE;                BRANCH IF IN RECEIVE MODE
MOVE.BMBSR,-(A7);            PUSH 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)+, MBDR;           TRANSMIT NEXT BYTE OF DATA
```

### 12.6.4 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 is an example showing how a master transmitter generates a stop condition.

```
MASTXMOVE.BMBSR, -(A7);  IF NO ACK, BRANCH TO END
BTST.B#0,(A7)+
BNE.B END
MOVE.BTXCNT,D0;          GET VALUE FROM THE TRANSMITTING COUNTER
BEQ.SEND;                IF NO MORE DATA, BRANCH TO END
MOVE.BDATABUF,-(A7);     TRANSMIT NEXT BYTE OF DATA
MOVE.B(A7)+,MBDR
MOVE.BTXCNT,D0;          DECREASE THE TXCNT
SUBQ.L#1,D0
MOVE.BD0,TXCNT
BRA.SEMASTX;             EXIT
ENDLEA.LMBCR,-(A7);      GENERATE A STOP CONDITION
BCLR.B#5,(A7)+
EMASTXRTE;               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 master receiver generates a STOP signal.

```
MASRMOVE.BRXCNT,D0;    DECREASE RXCNT
SUBQ.L#1,D0
MOVE.BD0,RXCNT
BEQ.SENMASR;           LAST BYTE TO BE READ
MOVE.BRXCNT,D1;        CHECK SECOND LAST BYTE TO BE READ (NOT LAST ONE OR SECOND
                       LAST
EXTB>LD1
SUBI.L#1,D1;
BNE.SNXMAR
LAMARBSET.B#3,MBCR;    SECOND LAST, DISABLE ACK TRANSMITTING
BRANXMAR
```

```
ENMASRBCLR.B#5,MBCR;   LAST ONE, GENERATE 'STOP' SIGNAL
NXMARMOVE.BMBDR,RXBUF; READ DATA AND STORE RTE
```

### 12.6.5 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 is as shown.

```
RESTARTMOVE.BMBCR,-(A7); ANOTHER START (RESTART)
BSET.B#2, (A7)
MOVE.B(A7)+, MBCR
MOVE.BCALLING,-(A7);     TRANSMIT THE CALLING ADDRESS, D0=R/W-
MOVE.BCALLING,-(A7);
MOVE.B(A7)+, MBDR
```

### 12.6.6 Slave Mode

In the slave interrupt service routine, the module addressed as slave bit (MAAS) should be tested to check if a calling of its own address has just been received. If MAAS is set, software should set the transmit/receive mode select bit (MTX bit of MBCR) according to the R/$\overline{W}$ command bit (SRW). Writing to the MBCR clears the MAAS automatically. The only time MAAS 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 MAAS cleared. A data transfer can now be initiated by writing information to MBDR, for slave transmits, or dummy reading from MBDR, in slave-receive mode. The slave will drive SCL low in between byte transfers. SCL is released when the MBDR 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.

### 12.6.7 Arbitration Lost

If several masters try to simultaneously engage the bus, only one master wins 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 MAL=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 MAL to indicate the failed attempt to engage the bus. When considering these cases, the slave service routine should test the MAL first and the software should clear the MAL bit if it is set.

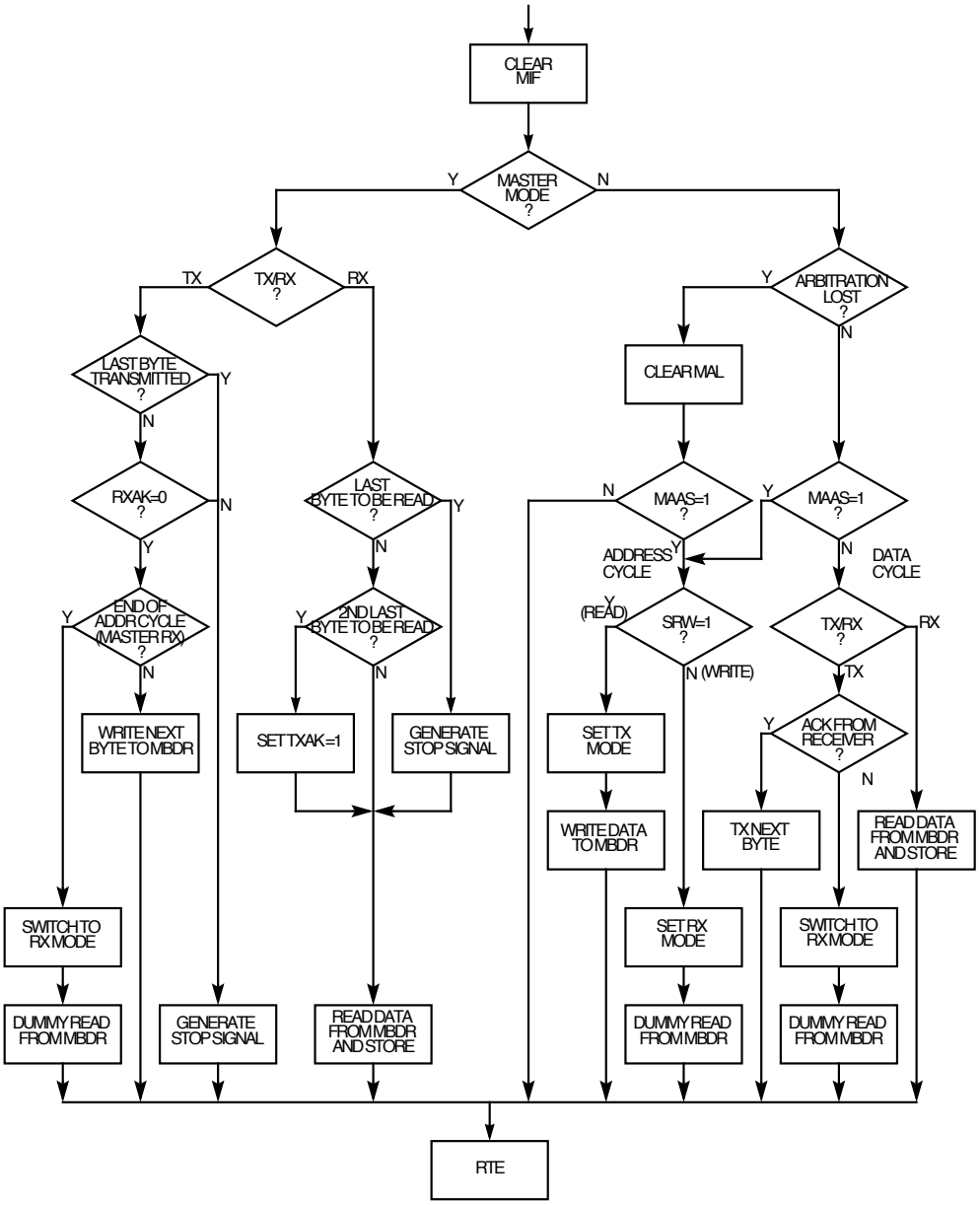


**Figure 12-4. Flow-Chart of Typical M-Bus Interrupt Routine**