43. Two-wire Interface (TWIHS)

43.1 Description

The Atmel Two-wire Interface (TWIHS) interconnects components on a unique two-wire bus, made up of one clock line and one data line with speeds of up to 400 kbit/s in Fast mode and up to 3.4 Mbit/s in High-speed slave mode only, based on a byte-oriented transfer format. It can be used with any Atmel Two-wire Interface bus Serial EEPROM and I²C-compatible devices, such as a Real-Time Clock (RTC), Dot Matrix/Graphic LCD Controller and temperature sensor. The TWIHS is programmable as a master or a slave with sequential or single-byte access. Multiple master capability is supported.

A configurable baud rate generator permits the output data rate to be adapted to a wide range of core clock frequencies.

Table 43-1 lists the compatibility level of the Atmel Two-wire Interface in Master mode and a full I^2C compatible device.

Table 43-1. Atmel TWI Compatibility with I^2C Standard

I^2C Standard	Atmel TWI
Standard Mode Speed (100 kHz)	Supported
Fast Mode Speed (400 kHz)	Supported
High-speed Mode (Slave only, 3.4 MHz)	Supported
7- or 10-bit(1) Slave Addressing	Supported
START Byte(2)	Not Supported
Repeated Start (Sr) Condition	Supported
ACK and NACK Management	Supported
Input Filtering	Supported
Slope Control	Not Supported
Clock Stretching	Supported
Multi Master Capability	Supported

Notes: 1. 10-bit support in Master mode only
2. START + b000000001 + Ack + Sr

43.2 Embedded Characteristics

- 3 TWIHSs
- Compatible with Atmel Two-wire Interface Serial Memory and I²C Compatible Devices(1)
- One, Two or Three Bytes for Slave Address
- Sequential Read/Write Operations
- Master and Multimaster Operation (Standard and Fast Modes Only)
- Slave Mode Operation (Standard, Fast and High-Speed Modes)
- Bit Rate: Up to 400 Kbit/s in Fast Mode and 3.4 Mbit/s in High-Speed Mode (Slave Mode Only)
- General Call Supported in Slave Mode
- SleepWalking (Asynchronous and Partial Wakeup)
- SMBus Support
- Connection to DMA Controller (DMA) Channel Capabilities Optimizes Data Transfers
- Register Write Protection

Note: 1. See Table 43-1 for details on compatibility with I²C Standard.

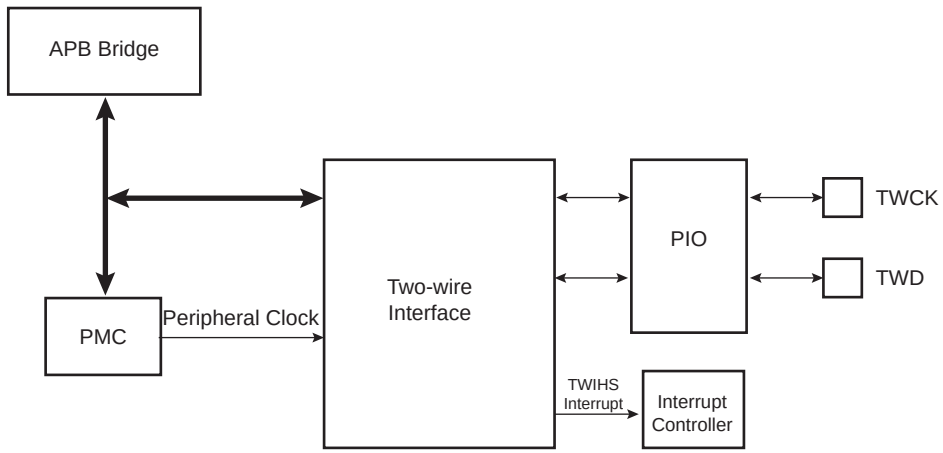
43.3 List of Abbreviations

Table 43-2. Abbreviations

Abbreviation	Description
TWI	Two-wire Interface
A	Acknowledge
NA	Non Acknowledge
P	Stop
S	Start
Sr	Repeated Start
SADR	Slave Address
ADR	Any address except SADR
R	Read
W	Write

43.4 Block Diagram

Figure 43-1. Block Diagram



43.4.1 I/O Lines Description

Table 43-3. I/O Lines Description

Pin Name	Pin Description	Type
TWD	Two-wire Serial Data	Input/Output
TWCK	Two-wire Serial Clock	Input/Output

43.5 Product Dependencies

43.5.1 I/O Lines

Both TWD and TWCK are bidirectional lines, connected to a positive supply voltage via a current source or pull-up resistor. When the bus is free, both lines are high. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function.

TWD and TWCK pins may be multiplexed with PIO lines. To enable the TWIHS, the user must program the PIO Controller to dedicate TWD and TWCK as peripheral lines. When High-speed Slave mode is enabled, the analog pad filter must be enabled.

The user must not program TWD and TWCK as open-drain. This is already done by the hardware.

Table 43-4. I/O Lines

Instance	Signal	I/O Line	Peripheral
TWIHS0	TWCK0	PA4	A
TWIHS0	TWD0	PA3	A
TWIHS1	TWCK1	PB5	A
TWIHS1	TWD1	PB4	A
TWIHS2	TWCK2	PD28	C
TWIHS2	TWD2	PD27	C

43.5.2 Power Management

Enable the peripheral clock.

The TWIHS may be clocked through the Power Management Controller (PMC), thus the user must first configure the PMC to enable the TWIHS clock.

43.5.3 Interrupt Sources

The TWIHS has an interrupt line connected to the Interrupt Controller. In order to handle interrupts, the Interrupt Controller must be programmed before configuring the TWIHS.

Table 43-5. Peripheral IDs

Instance	ID
TWIHS0	19
TWIHS1	20
TWIHS2	41

43.6 Functional Description

43.6.1 Transfer Format

The data put on the TWD line must be 8 bits long. Data is transferred MSB first; each byte must be followed by an acknowledgement. The number of bytes per transfer is unlimited. See Figure 43-3.

Each transfer begins with a START condition and terminates with a STOP condition. See Figure 43-2.

- A high-to-low transition on the TWD line while TWCK is high defines the START condition.
- A low-to-high transition on the TWD line while TWCK is high defines the STOP condition.

Figure 43-2. START and STOP Conditions

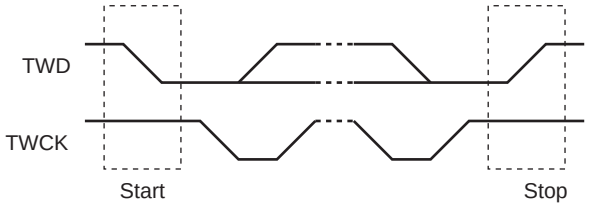
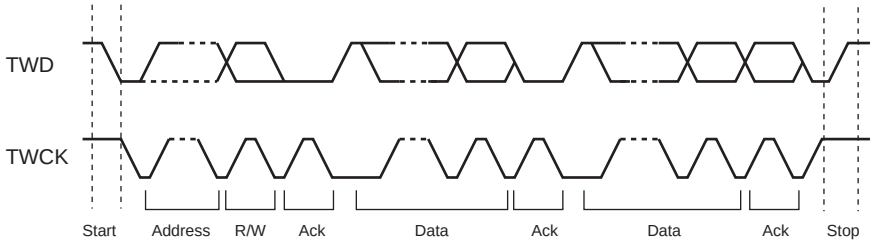


Figure 43-3. Transfer Format



43.6.2 Modes of Operation

The TWIHS has different modes of operation:

- Master Transmitter mode (Standard and Fast modes only)
- Master Receiver mode (Standard and Fast modes only)
- Multimaster Transmitter mode (Standard and Fast modes only)
- Multimaster Receiver mode (Standard and Fast modes only)
- Slave Transmitter mode (Standard, Fast and High-speed modes)
- Slave Receiver mode (Standard, Fast and High-speed modes)

These modes are described in the following sections.

43.6.3 Master Mode

43.6.3.1 Definition

The master is the device that starts a transfer, generates a clock and stops it. This operating mode is not available if High-speed mode is selected.

43.6.3.2 Programming Master Mode

The following registers must be programmed before entering Master mode:

1. TWIHS_MMR.DADR (+ IADRSZ + IADR if a 10-bit device is addressed): The device address is used to access slave devices in Read or Write mode.
2. TWIHS_CWGR.CKDIV + CHDIV + CLDIV: Clock Waveform register
3. TWIHS_CR.SVDIS: Disables the Slave mode
4. TWIHS_CR.MSEN: Enables the Master mode

Note: If the TWIHS is already in Master mode, the device address (DADR) can be configured without disabling the Master mode.

43.6.3.3 Master Transmitter Mode

This operating mode is not available if High-speed mode is selected.

After the master initiates a START condition when writing into the Transmit Holding register (TWIHS_THR), it sends a 7-bit slave address, configured in the Master Mode register (DADR in TWIHS_MMR), to notify the slave device. The bit following the slave address indicates the transfer direction, 0 in this case (MREAD = 0 in TWIHS_MMR).

The TWIHS transfers require the slave to acknowledge each received byte. During the acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. If the slave does not acknowledge the byte, then the Not Acknowledge flag (NACK) is set in the TWIHS Status Register (TWIHS_SR) of the master and a STOP condition is sent. The NACK flag must be cleared by reading TWIHS_SR before the next write into TWIHS_THR. As with the other status bits, an interrupt can be generated if enabled in the Interrupt Enable register (TWIHS_IER). If the slave acknowledges the byte, the data written in the TWIHS_THR is then shifted in the internal shifter and transferred. When an acknowledge is detected, the TXRDY bit is set until a new write in the TWIHS_THR.

TXRDY is used as Transmit Ready for the DMA transmit channel.

While no new data is written in the TWIHS_THR, the serial clock line is tied low. When new data is written in the TWIHS_THR, the SCL is released and the data is sent. Setting the STOP bit in TWIHS_CR generates a STOP condition.

After a master write transfer, the serial clock line is stretched (tied low) as long as no new data is written in the TWIHS_THR or until a STOP command is performed.

To clear the TXRDY flag, first set the bit TWIHS_CR.MSDIS, then set the bit TWIHS_CR.MSEN.

See Figure 43-4, Figure 43-5, and Figure 43-6.

Figure 43-4. Master Write with One Data Byte

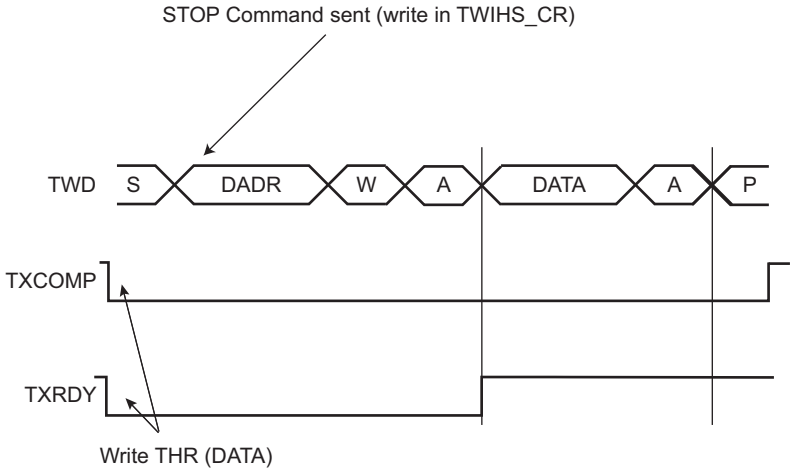


Figure 43-5. Master Write with Multiple Data Bytes

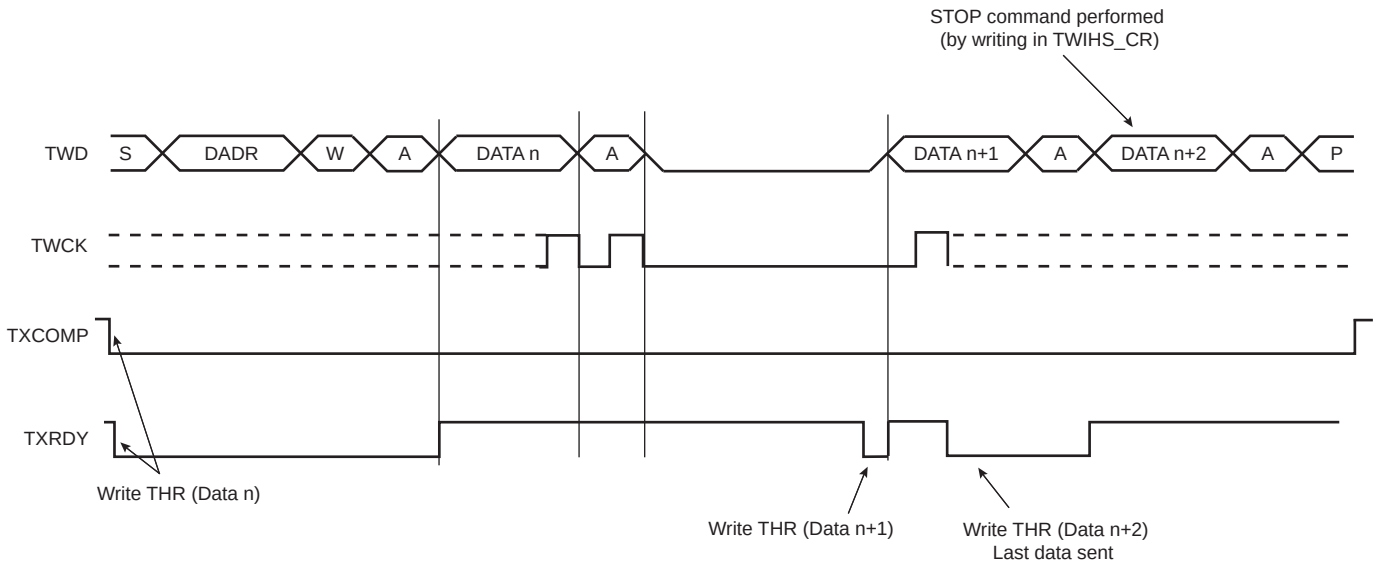
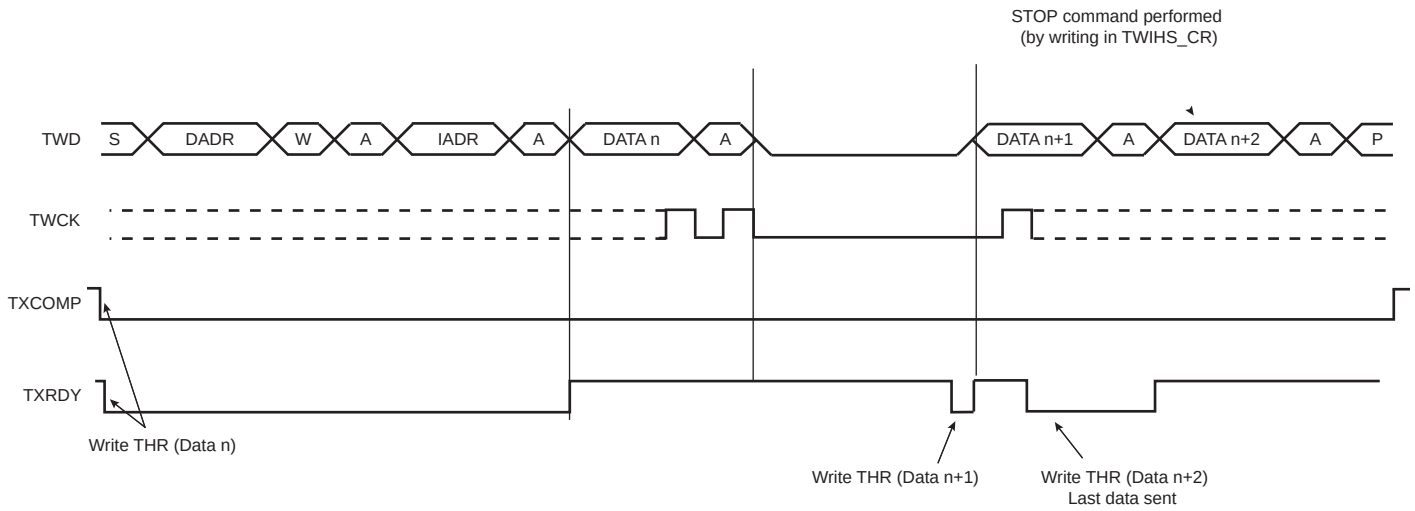


Figure 43-6. Master Write with One-Byte Internal Address and Multiple Data Bytes



43.6.3.4 Master Receiver Mode

Master Receiver mode is not available if High-speed mode is selected.

The read sequence begins by setting the START bit. After the START condition has been sent, the master sends a 7-bit slave address to notify the slave device. The bit following the slave address indicates the transfer direction, 1 in this case (MREAD = 1 in TWIHS_MMR). During the acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. The master polls the data line during this clock pulse and sets the NACK bit in the TWIHS_SR if the slave does not acknowledge the byte.

If an acknowledge is received, the master is then ready to receive data from the slave. After data has been received, the master sends an acknowledge condition to notify the slave that the data has been received except for the last data (see Figure 43-7). When the RXRDY bit is set in the TWIHS_SR, a character has been received in the Receive Holding register (TWIHS_RHR). The RXRDY bit is reset when reading the TWIHS_RHR.

When a single data byte read is performed, with or without internal address **(**IADR), the START and STOP bits must be set at the same time. See Figure 43-7. When a multiple data byte read is performed, with or without internal address **(**IADR), the STOP bit must be set after the next-to-last data received (same condition applies for START bit to generate a REPEATED START). See Figure 43-8. For internal address usage, see Section 43.6.3.5 ”Internal Address”.

If TWIHS_RHR is full (RXRDY high) and the master is receiving data, the serial clock line is tied low before receiving the last bit of the data and until the TWIHS_RHR is read. Once the TWIHS_RHR is read, the master stops stretching the serial clock line and ends the data reception. See Figure 43-9.

Warning: When receiving multiple bytes in Master Read mode, if the next-to-last access is not read (the RXRDY flag remains high), the last access is not completed until TWIHS_RHR is read. The last access stops on the next-to-last bit (clock stretching). When the TWIHS_RHR is read, there is only half a bit period to send the STOP (or START) command, else another read access might occur (spurious access).

A possible workaround is to set the STOP (or START) bit before reading the TWIHS_RHR on the next-to-last access (within IT handler).

Figure 43-7. Master Read with One Data Byte

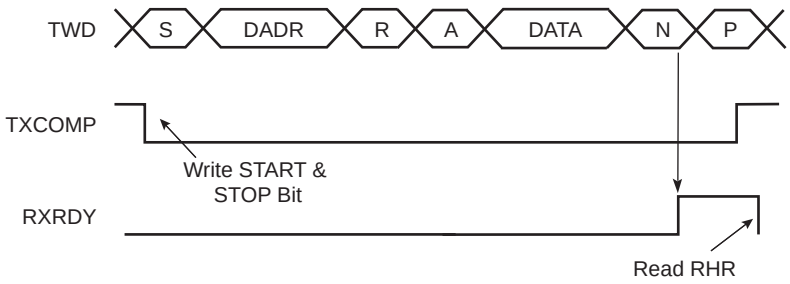


Figure 43-8. Master Read with Multiple Data Bytes

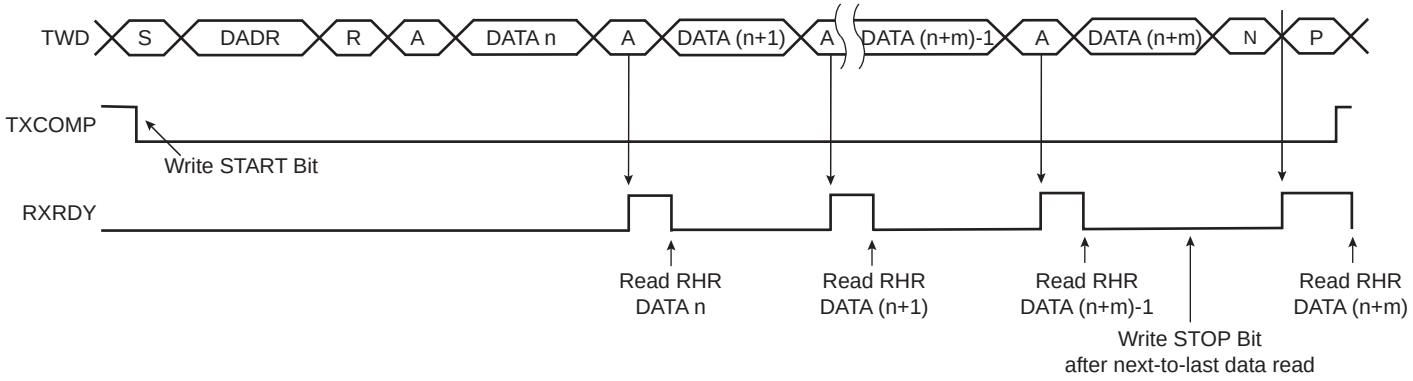
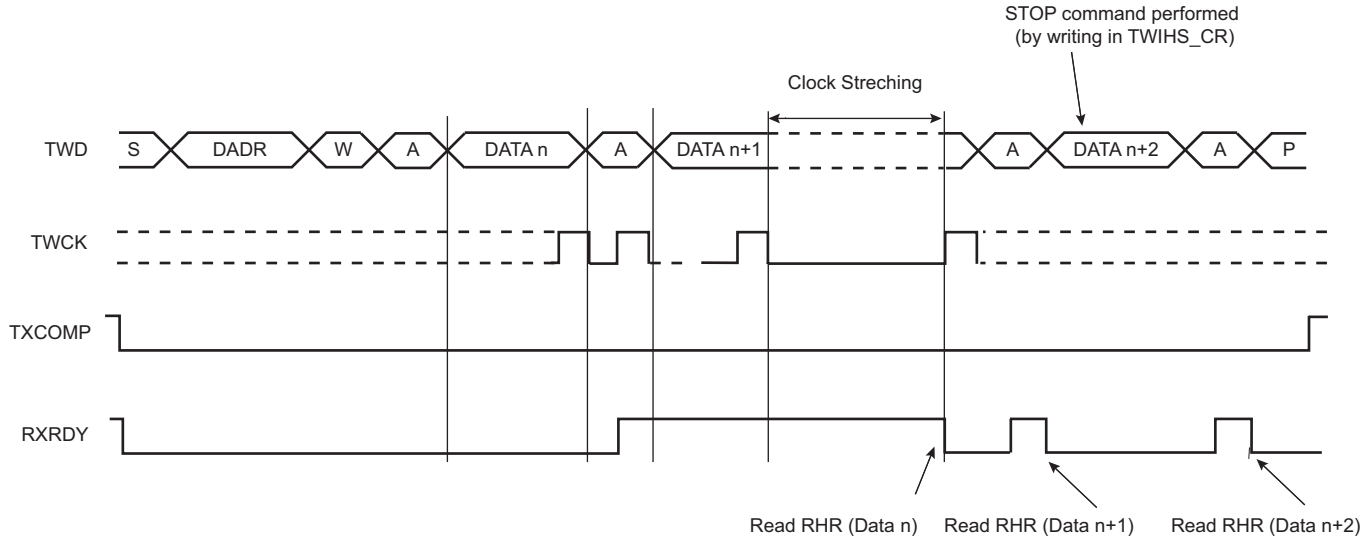


Figure 43-9. Master Read Clock Stretching with Multiple Data Bytes



RXRDY is used as receive ready for the DMA receive channel.

43.6.3.5 Internal Address

The TWIHS can perform transfers with 7-bit slave address devices and with 10-bit slave address devices.

7-bit Slave Addressing

When addressing 7-bit slave devices, the internal address bytes are used to perform random address (read or write) accesses to reach one or more data bytes, e.g. within a memory page location in a serial memory. When performing read operations with an internal address, the TWIHS performs a write operation to set the internal address into the slave device, and then switch to Master Receiver mode. Note that the second START condition (after sending the IADR) is sometimes called “repeated start” (Sr) in I^2C fully-compatible devices. See Figure 43-11.

See Figure 43-10 and Figure 43-12 for the master write operation with internal address.

The three internal address bytes are configurable through TWIHS_MMR.

If the slave device supports only a 7-bit address, i.e., no internal address, IADRSZ must be set to 0.

Table 43-6 shows the abbreviations used in Figure 43-10 and Figure 43-11.

Table 43-6. Abbreviations

Abbreviation	Definition
S	Start
Sr	Repeated Start
P	Stop
W	Write
R	Read
A	Acknowledge
NA	Not Acknowledge
DADR	Device Address
IADR	Internal Address

Figure 43-10. Master Write with One-, Two- or Three-Byte Internal Address and One Data Byte

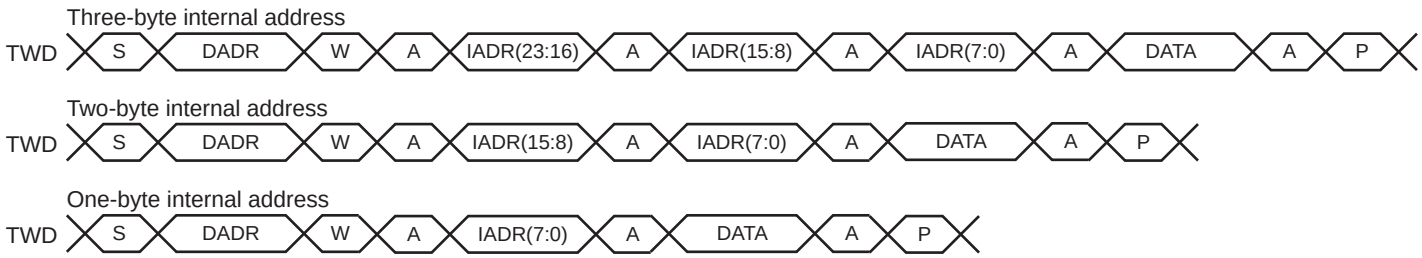
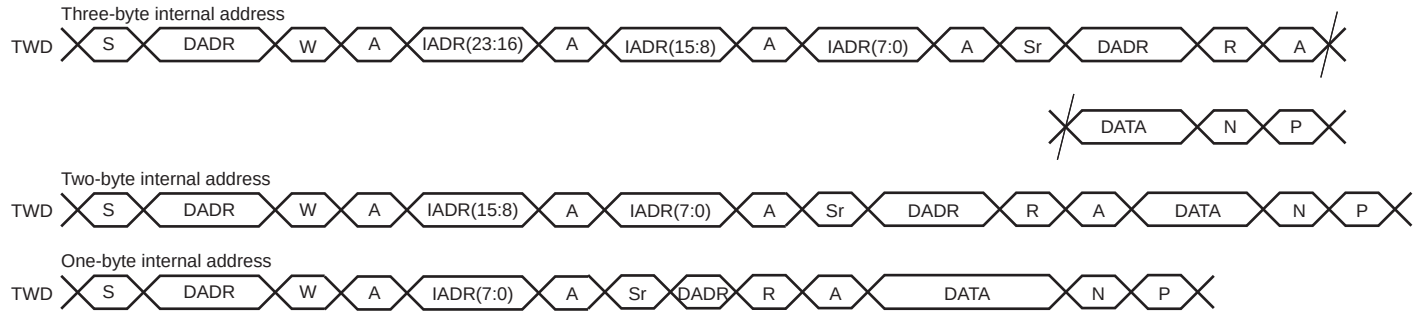


Figure 43-11. Master Read with One-, Two- or Three-Byte Internal Address and One Data Byte



10-bit Slave Addressing

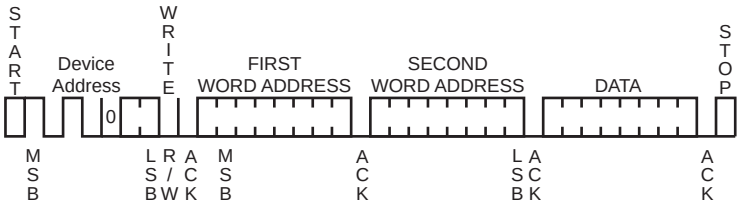
For a slave address higher than seven bits, configure the address size (IADRSZ) and set the other slave address bits in the Internal Address register (TWIHS_IADR). The two remaining internal address bytes, IADR[15:8] and IADR[23:16], can be used the same way as in 7-bit slave addressing.

Example: Address a 10-bit device (10-bit device address is b1 b2 b3 b4 b5 b6 b7 b8 b9 b10)

1. Program IADRSZ = 1,
2. Program DADR with 1 1 1 1 0 b1 b2 (b1 is the MSB of the 10-bit address, b2, etc.)
3. Program TWIHS_IADR with b3 b4 b5 b6 b7 b8 b9 b10 (b10 is the LSB of the 10-bit address)

Figure 43-12 shows a byte write to a memory device. This demonstrates the use of internal addresses to access the device.

Figure 43-12. Internal Address Usage



43.6.3.6 Repeated Start

In addition to Internal Address mode, REPEATED START (Sr) can be generated manually by writing the START bit at the end of a transfer instead of the STOP bit. In such case, the parameters of the next transfer (direction, SADR, etc.) need to be set before writing the START bit at the end of the previous transfer.

See Section 43.6.3.11 "Read/Write Flowcharts" for detailed flowcharts.

Note that generating a REPEATED START after a single data read is not supported.

43.6.3.7 Bus Clear Command

The TWIHS can perform a Bus Clear command:

1. Configure the Master mode (DADR, CKDIV, etc).
2. Start the transfer by setting the CLEAR bit in the TWIHS_CR.

43.6.3.8 Using the DMA Controller (DMAC) in Master Mode

The use of the DMA significantly reduces the CPU load.

To ensure correct implementation, follow the programming sequences below:

Data Transmit with the DMA in Master Mode

The DMA transfer size must be defined with the buffer size minus 1. The remaining character must be managed without DMA to ensure that the exact number of bytes are transmitted regardless of system bus latency conditions during the end of the buffer transfer period.

1. Initialize the DMA (channels, memory pointers, size - 1, etc.);
2. Configure the Master mode (DADR, CKDIV, MREAD = 0, etc.) or Slave mode.
3. Enable the DMA.
4. Wait for the DMA status flag indicating that the buffer transfer is complete.
5. Disable the DMA.
6. Wait for the TXRDY flag in TWIHS_SR.
7. Set the STOP bit in TWIHS_CR.
8. Write the last character in TWIHS_THR.
9. (Only if peripheral clock must be disabled) Wait for the TXCOMP flag to be raised in TWIHS_SR.

Data Receive with the DMA in Master Mode

The DMA transfer size must be defined with the buffer size minus 2. The two remaining characters must be managed without DMA to ensure that the exact number of bytes are received regardless of system bus latency conditions encountered during the end of buffer transfer period.

1. Initialize the DMA (channels, memory pointers, size - 2, etc.);
2. Configure the Master mode (DADR, CKDIV, MREAD = 1, etc.) or Slave mode.
3. Enable the DMA.
4. (Master Only) Write the START bit in the TWIHS_CR to start the transfer.
5. Wait for the DMA status flag indicating that the buffer transfer is complete.

6. Disable the DMA.
7. Wait for the RXRDY flag in the TWIHS_SR.
8. Set the STOP bit in TWIHS_CR.
9. Read the penultimate character in TWIHS_RHR.
10. Wait for the RXRDY flag in the TWIHS_SR.
11. Read the last character in TWIHS_RHR.
12. (Only if peripheral clock must be disabled) Wait for the TXCOMP flag to be raised in TWIHS_SR.

43.6.3.9 SMBus Mode

SMBus mode is enabled when a one is written to the SMEN bit in the TWIHS_CR. SMBus mode operation is similar to I²C operation with the following exceptions:

- Only 7-bit addressing can be used.
- The SMBus standard describes a set of timeout values to ensure progress and throughput on the bus. These timeout values must be programmed into TWIHS_SMBTR.
- Transmissions can optionally include a CRC byte, called Packet Error Check (PEC).
- A set of addresses has been reserved for protocol handling, such as alert response address (ARA) and host header (HH) address. Address matching on these addresses can be enabled by configuring the TWIHS_CR.

Packet Error Checking

Each SMBus transfer can optionally end with a CRC byte, called the PEC byte. Writing a one to the PECEN bit in TWIHS_CR enables automatic PEC handling in the current transfer. Transfers with and without PEC can be intermixed in the same system, since some slaves may not support PEC. The PEC LFSR is always updated on every bit transmitted or received, so that PEC handling on combined transfers is correct.

In Master Transmitter mode, the master calculates a PEC value and transmits it to the slave after all data bytes have been transmitted. Upon reception of this PEC byte, the slave compares it to the PEC value it has computed itself. If the values match, the data was received correctly, and the slave returns an ACK to the master. If the PEC values differ, data was corrupted, and the slave returns a NACK value. Some slaves may not be able to check the received PEC in time to return a NACK if an error occurred. In this case, the slave should always return an ACK after the PEC byte, and another method must be used to verify that the transmission was received correctly.

In Master Receiver mode, the slave calculates a PEC value and transmits it to the master after all data bytes have been transmitted. Upon reception of this PEC byte, the master compares it to the PEC value it has computed itself. If the values match, the data was received correctly. If the PEC values differ, data was corrupted, and the PECERR bit in TWIHS_SR is set. In Master Receiver mode, the PEC byte is always followed by a NACK transmitted by the master, since it is the last byte in the transfer.

In combined transfers, the PECRQ bit should only be set in the last of the combined transfers.

Consider the following transfer:

S, ADR+W, COMMAND_BYTE, ACK, SR, ADR+R, DATA_BYTE, ACK, PEC_BYTE, NACK, P

See Section 43.6.3.11 ”Read/Write Flowcharts” for detailed flowcharts.

Timeouts

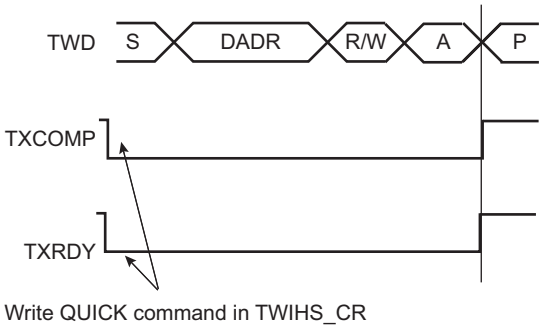
The TLOWS and TLOWM fields in TWIHS_SMBTR configure the SMBus timeout values. If a timeout occurs, the master transmits a STOP condition and leaves the bus. The TOUT bit is also set in TWIHS_SR.

43.6.3.10 SMBus Quick Command (Master Mode Only)

The TWIHS can perform a quick command:

1. Configure the Master mode (DADR, CKDIV, etc).
2. Write the MREAD bit in the TWIHS_MMR at the value of the one-bit command to be sent.
3. Start the transfer by setting the QUICK bit in the TWIHS_CR.

Figure 43-13. SMBus Quick Command



43.6.3.11 Read/Write Flowcharts

The flowcharts give examples for read and write operations. A polling or interrupt method can be used to check the status bits. The interrupt method requires that TWIHS_IER be configured first.

Figure 43-14. TWIHS Write Operation with Single Data Byte without Internal Address

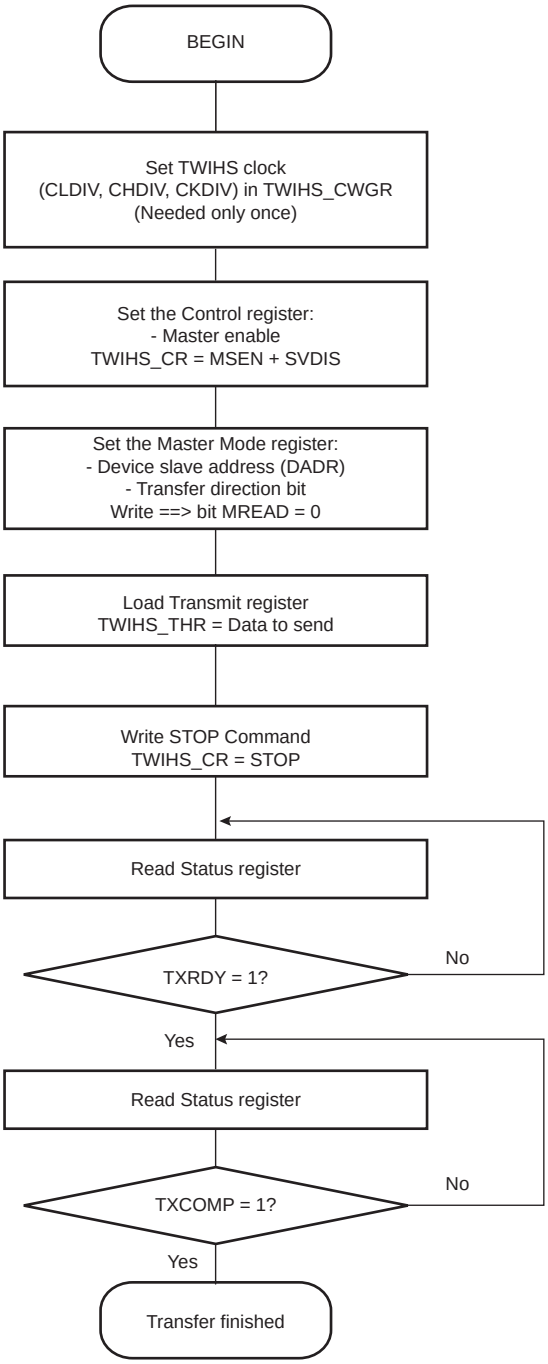


Figure 43-15. TWIHS Write Operation with Single Data Byte and Internal Address

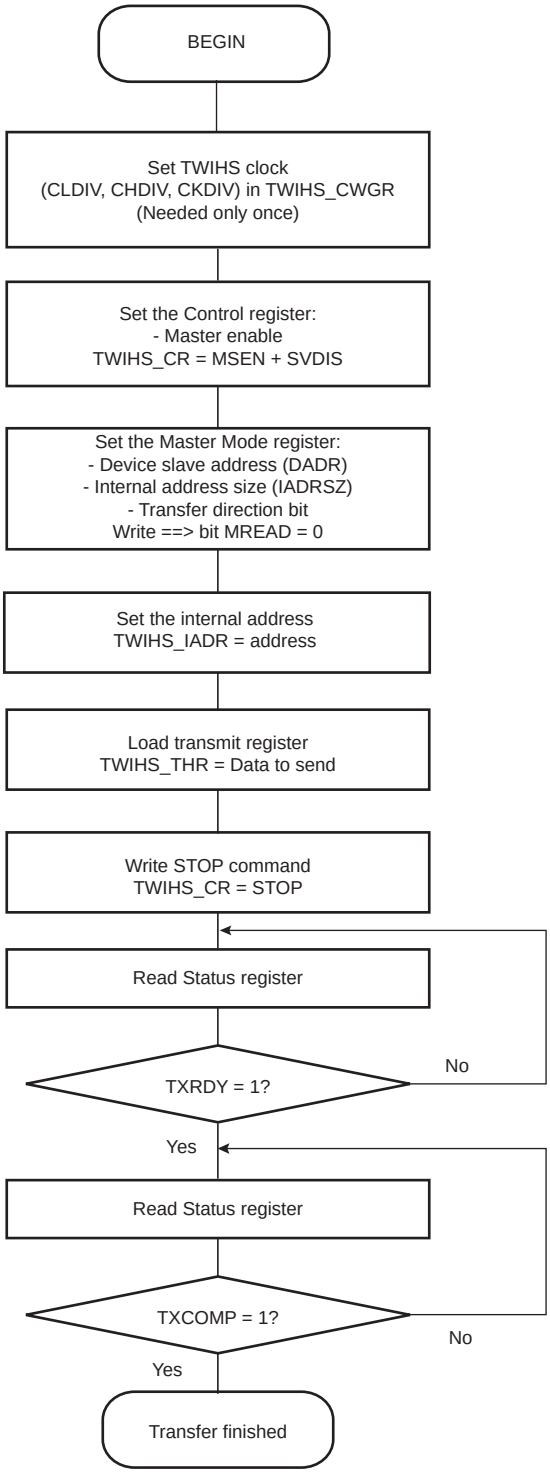


Figure 43-16. TWIHS Write Operation with Multiple Data Bytes with or without Internal Address

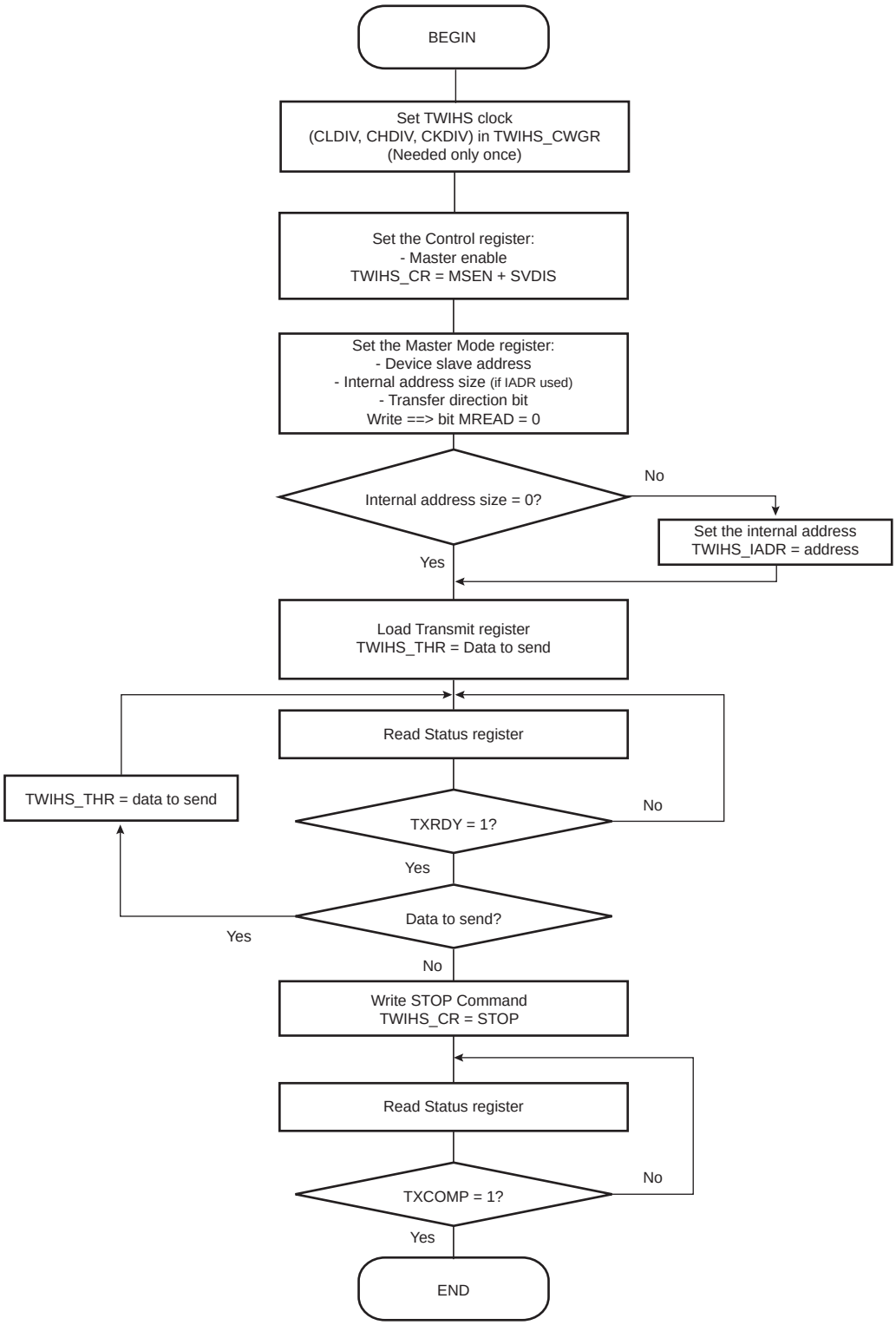


Figure 43-17. SMBus Write Operation with Multiple Data Bytes with or without Internal Address and PEC Sending

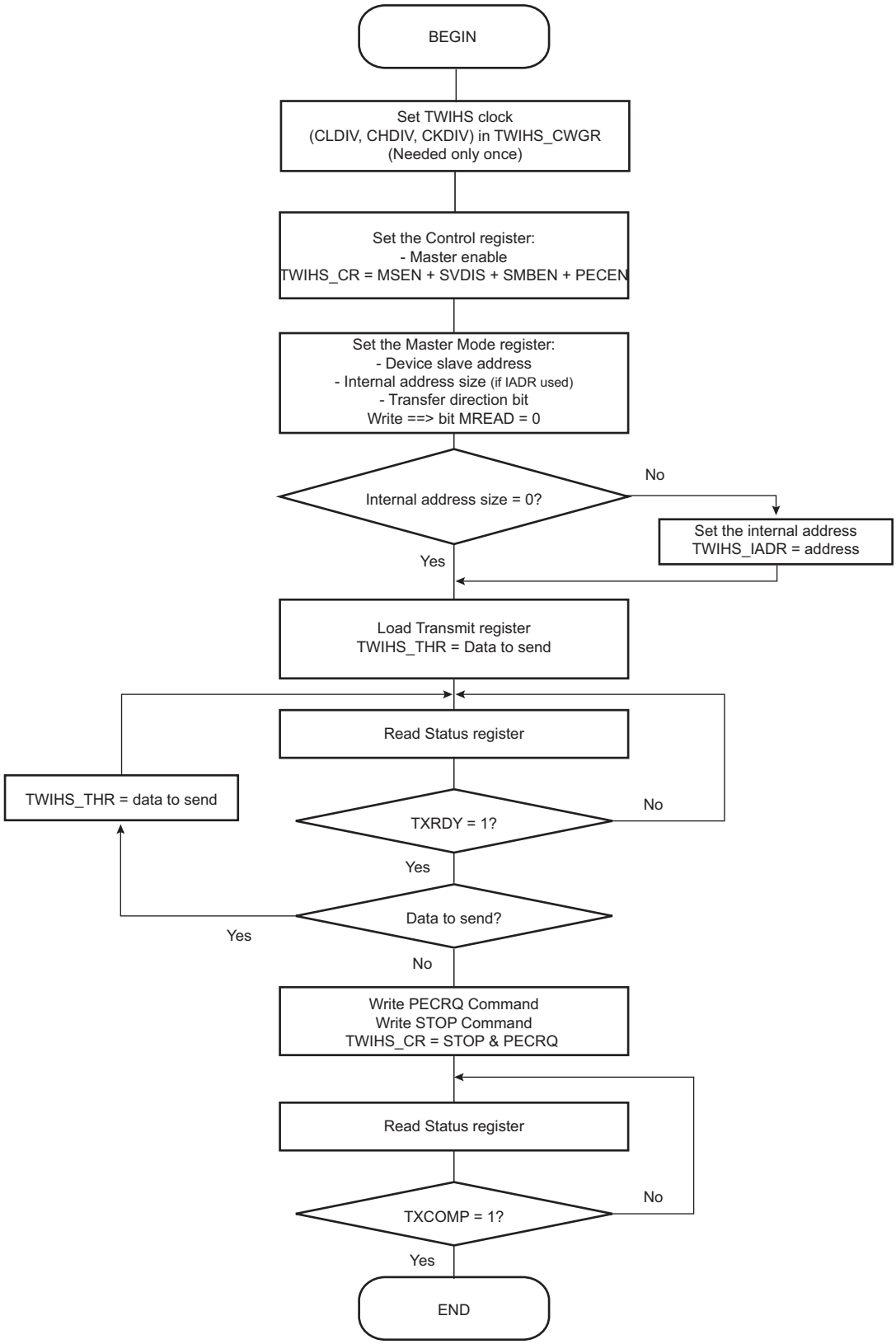


Figure 43-18. SMBus Write Operation with Multiple Data Bytes with PEC and Alternative Command Mode

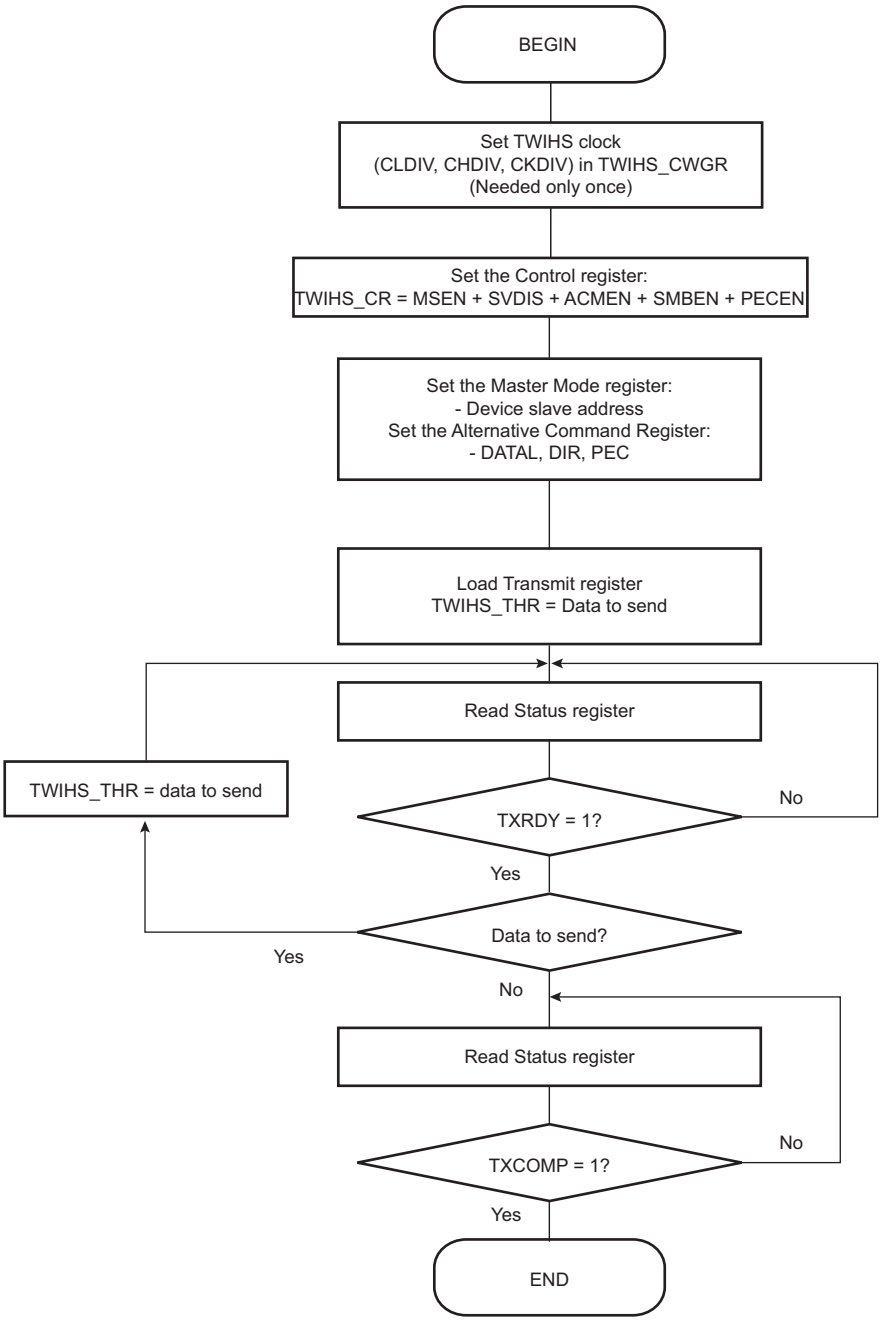


Figure 43-19. TWIHS Write Operation with Multiple Data Bytes and Read Operation with Multiple Data Bytes (Sr)

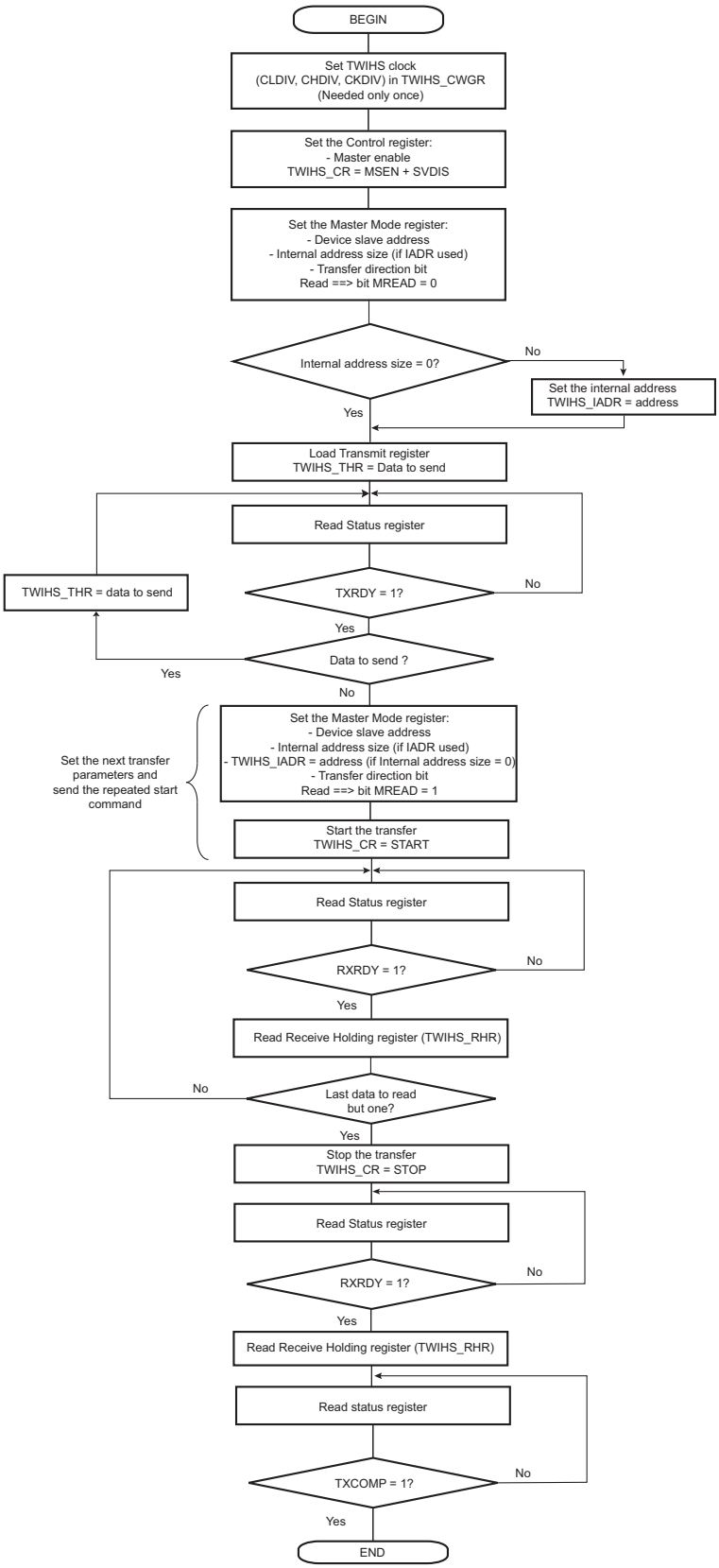


Figure 43-20. TWIHS Write Operation with Multiple Data Bytes + Read Operation and Alternative Command Mode + PEC

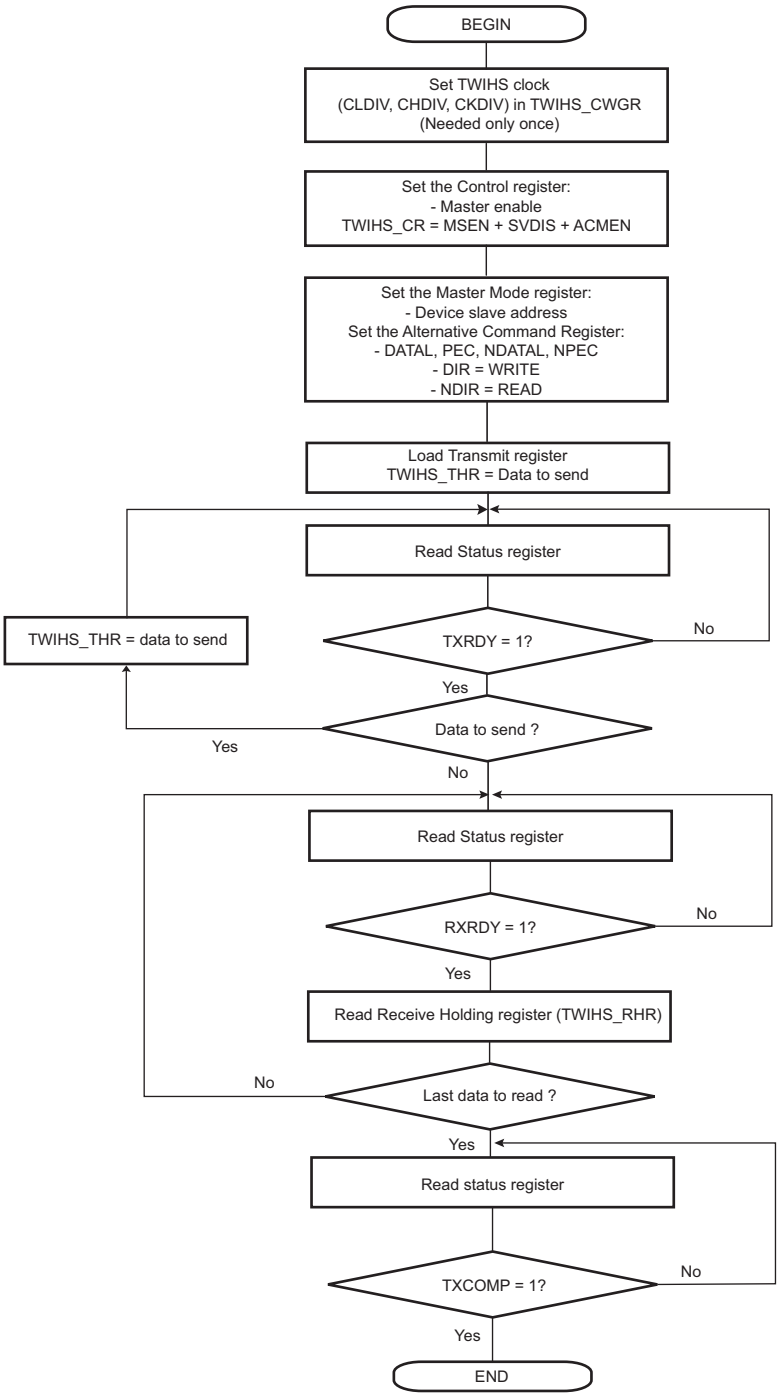


Figure 43-21. TWIHS Read Operation with Single Data Byte without Internal Address

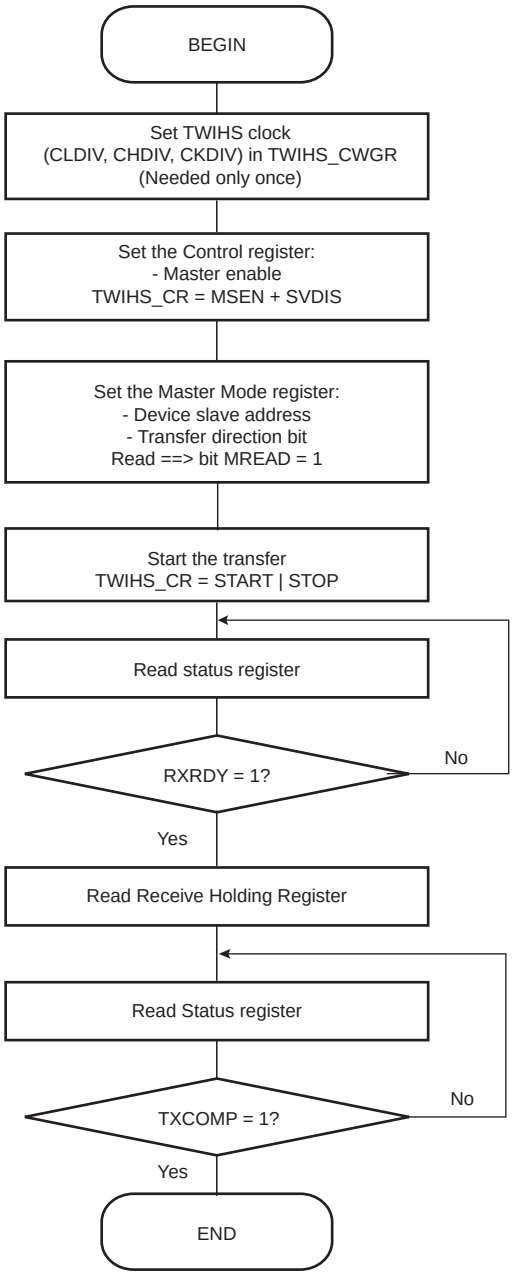


Figure 43-22. TWIHS Read Operation with Single Data Byte and Internal Address

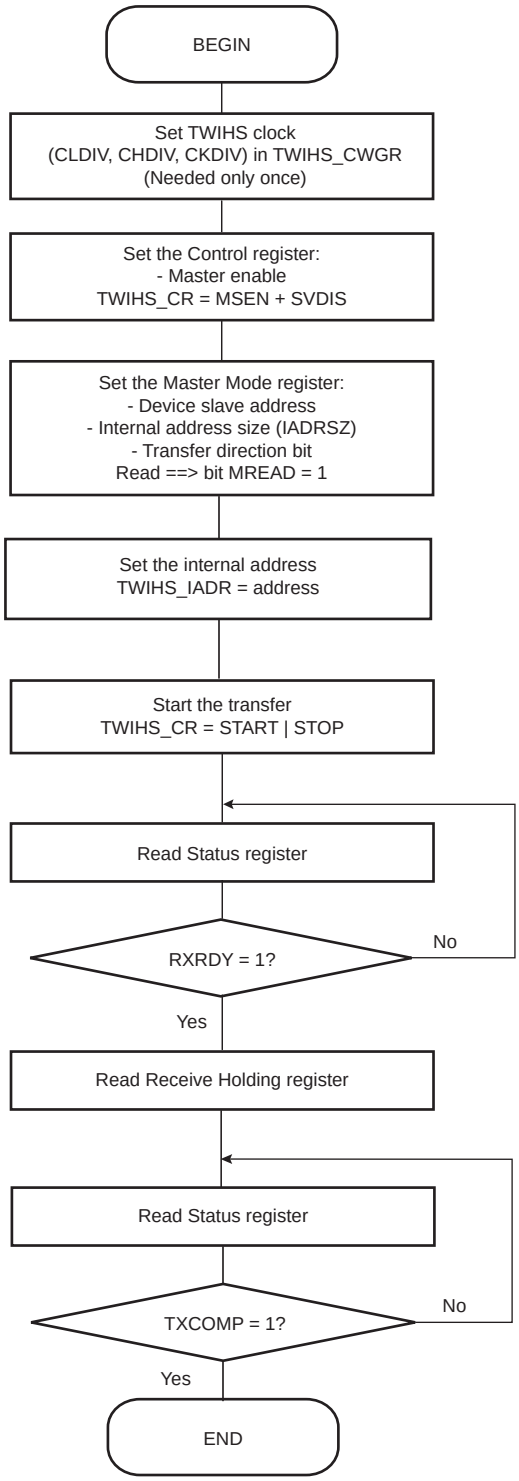


Figure 43-23. TWIHS Read Operation with Multiple Data Bytes with or without Internal Address

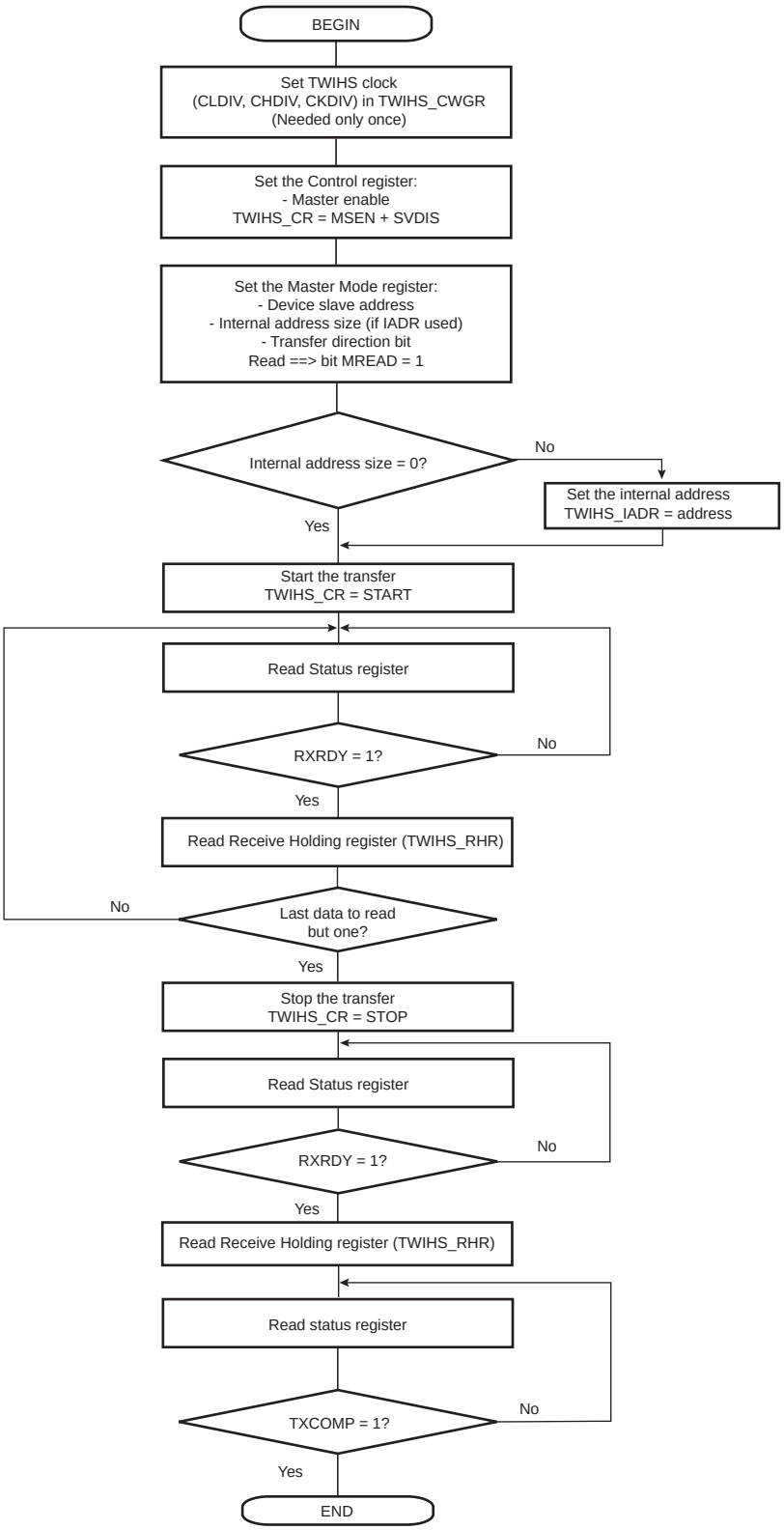


Figure 43-24. TWIHS Read Operation with Multiple Data Bytes with or without Internal Address with PEC

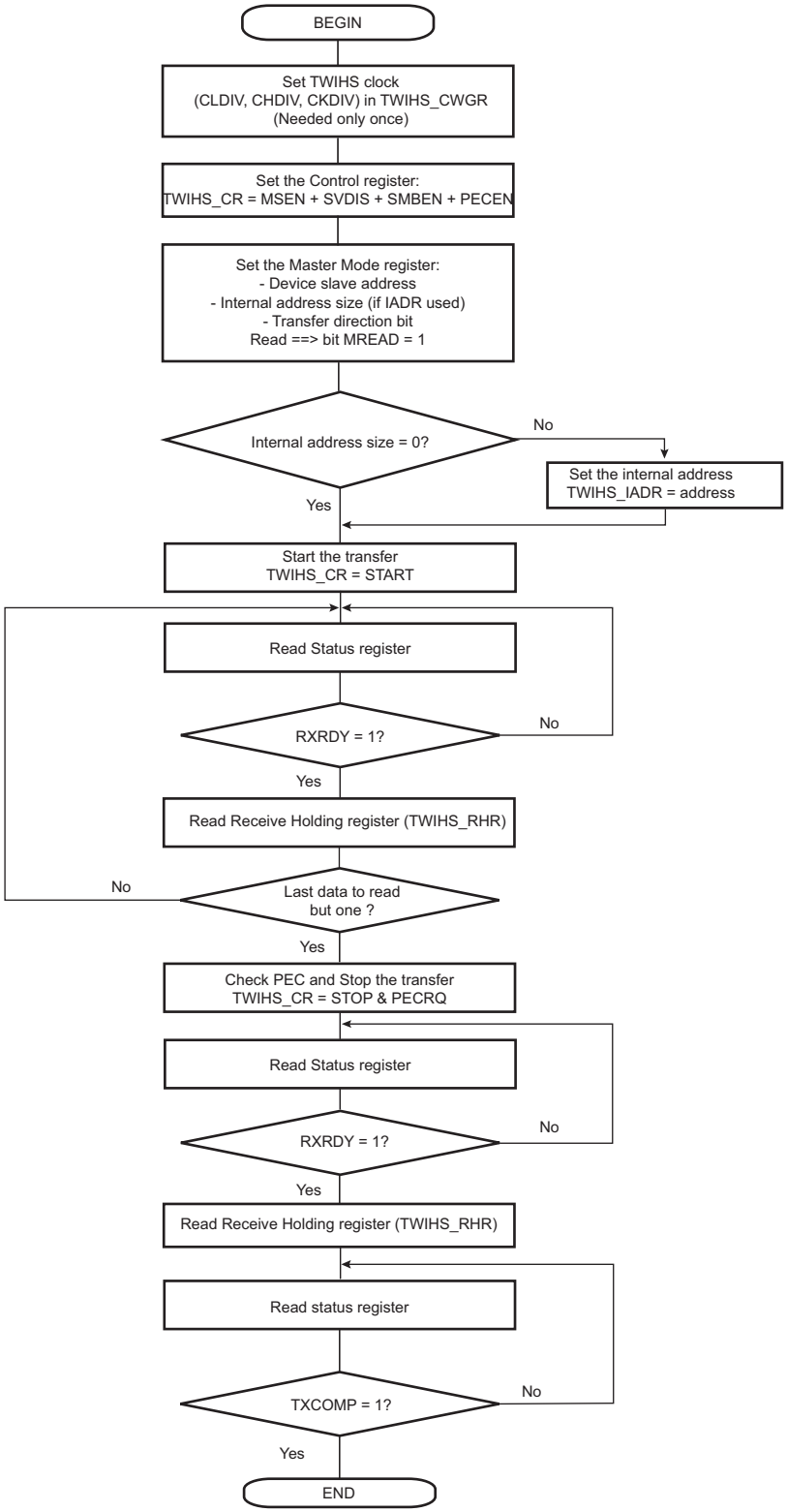


Figure 43-25. TWIHS Read Operation with Multiple Data Bytes with Alternative Command Mode with PEC

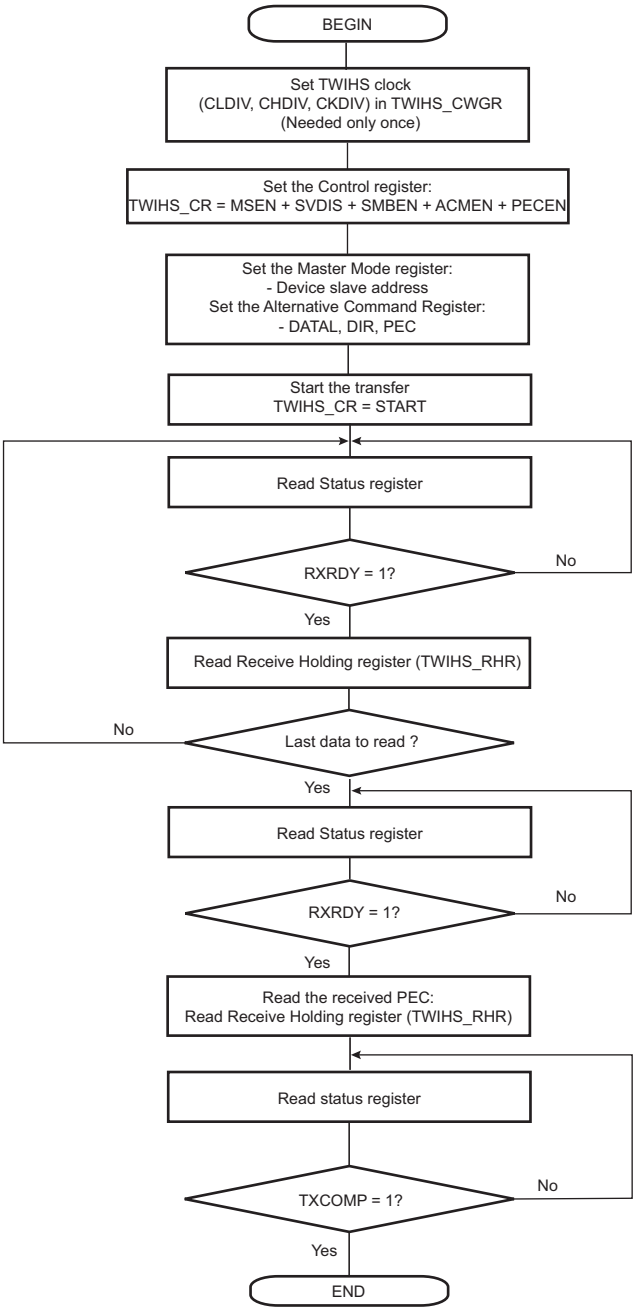


Figure 43-26. TWIHS Read Operation with Multiple Data Bytes + Write Operation with Multiple Data Bytes (Sr)

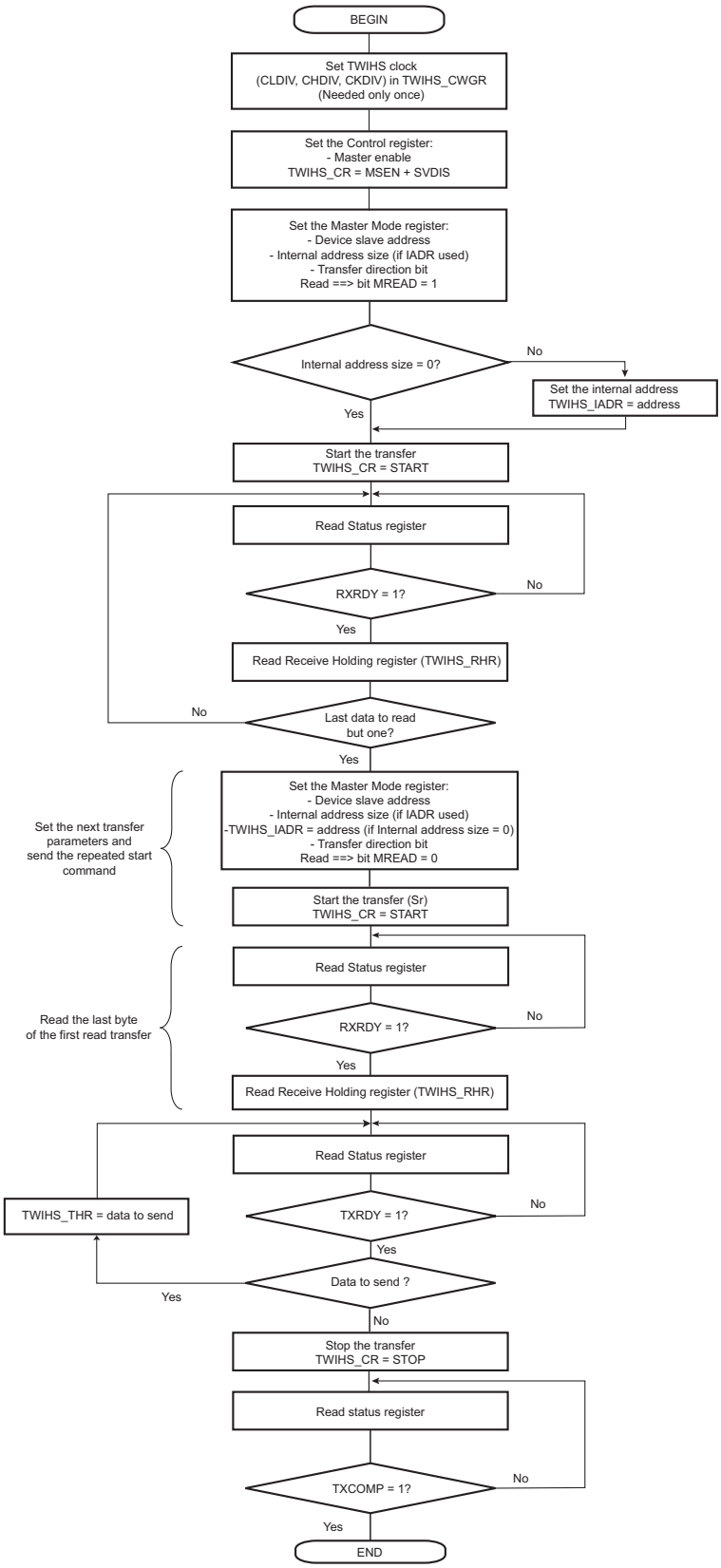
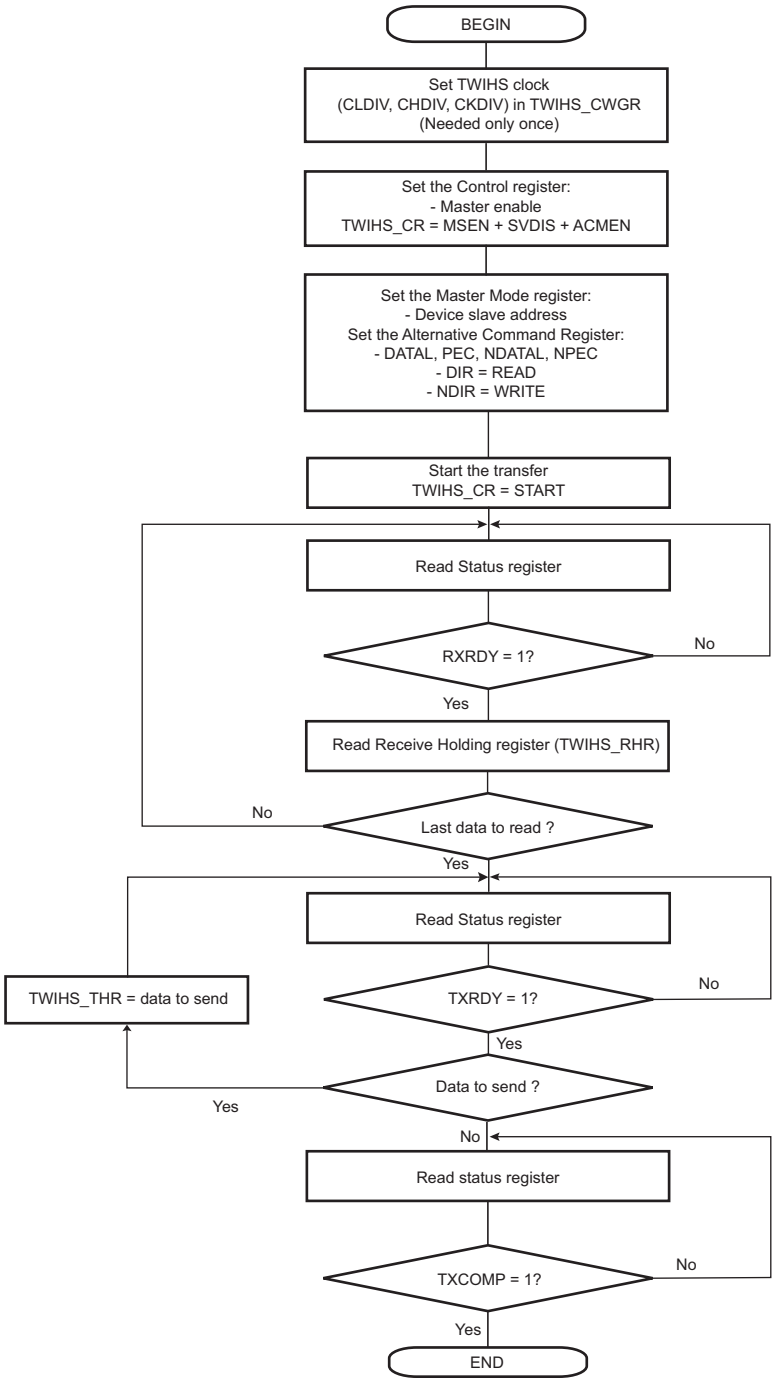


Figure 43-27. TWIHS Read Operation with Multiple Data Bytes + Write with Alternative Command Mode with PEC



43.6.4 Multimaster Mode

43.6.4.1 Definition

In Multimaster mode, more than one master may handle the bus at the same time without data corruption by using arbitration.

Arbitration starts as soon as two or more masters place information on the bus at the same time, and stops (arbitration is lost) for the master that intends to send a logical one while the other master sends a logical zero.

As soon as arbitration is lost by a master, it stops sending data and listens to the bus in order to detect a stop. When the stop is detected, the master that has lost arbitration may put its data on the bus by respecting arbitration.

Arbitration is illustrated in Figure 43-29.

43.6.4.2 Different Multimaster Modes

Two Multimaster modes may be distinguished:

1. The TWIHS is considered as a master only and is never addressed.
2. The TWIHS may be either a master or a slave and may be addressed.

Note: Arbitration in supported in both Multimaster modes.

TWIHS as Master Only

In this mode, the TWIHS is considered as a master only (MSEN is always at one) and must be driven like a master with the ARBLST (Arbitration Lost) flag in addition.

If arbitration is lost (ARBLST = 1), the user must reinitiate the data transfer.

If starting a transfer (ex.: DADR + START + W + Write in THR) and if the bus is busy, the TWIHS automatically waits for a STOP condition on the bus to initiate the transfer (see Figure 43-28).

Note: The state of the bus (busy or free) is not indicated in the user interface.

TWIHS as Master or Slave

The automatic reversal from master to slave is not supported in case of a lost arbitration.

Then, in the case where TWIHS may be either a master or a slave, the user must manage the pseudo Multimaster mode described in the steps below:

1. Program the TWIHS in Slave mode (SADR + MSDIS + SVEN) and perform a slave access (if TWIHS is addressed).
2. If the TWIHS has to be set in Master mode, wait until TXCOMP flag is at 1.
3. Program the Master mode (DADR + SVDIS + MSEN) and start the transfer (ex: START + Write in THR).
4. As soon as the Master mode is enabled, the TWIHS scans the bus in order to detect if it is busy or free. When the bus is considered free, the TWIHS initiates the transfer.
5. As soon as the transfer is initiated and until a STOP condition is sent, the arbitration becomes relevant and the user must monitor the ARBLST flag.
6. If the arbitration is lost (ARBLST is set to 1), the user must program the TWIHS in Slave mode in case the master that won the arbitration needs to access the TWIHS.
7. If the TWIHS has to be set in Slave mode, wait until the TXCOMP flag is at 1 and then program the Slave mode.

Note: If the arbitration is lost and the TWIHS is addressed, the TWIHS does not acknowledge, even if it is programmed in Slave mode as soon as ARBLST is set to 1. Then the master must repeat SADR.

Figure 43-28. User Sends Data While the Bus is Busy

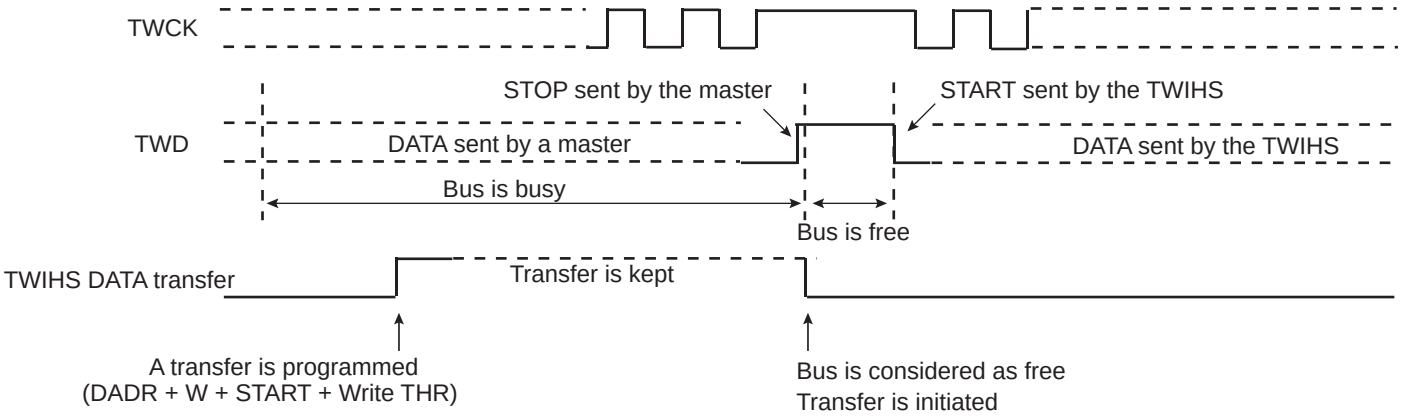
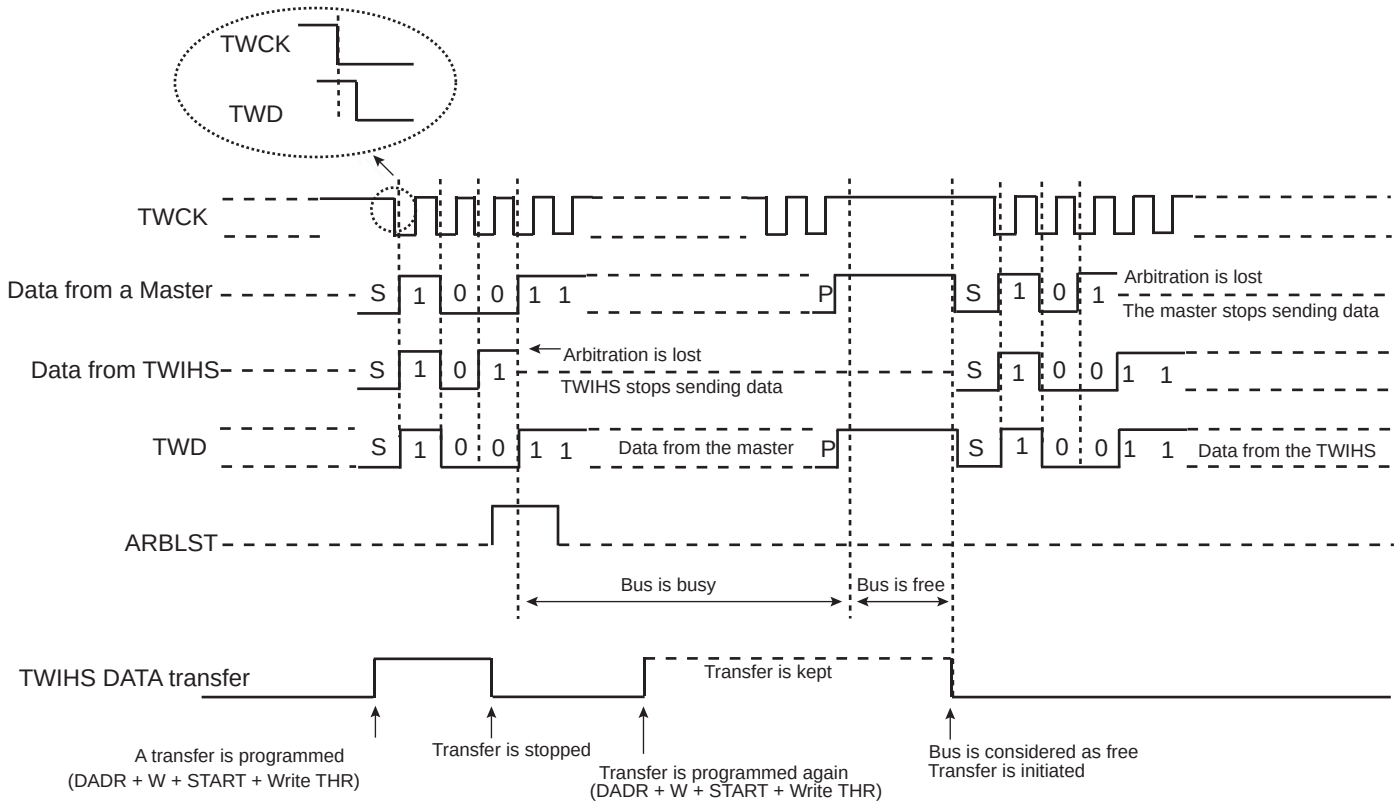
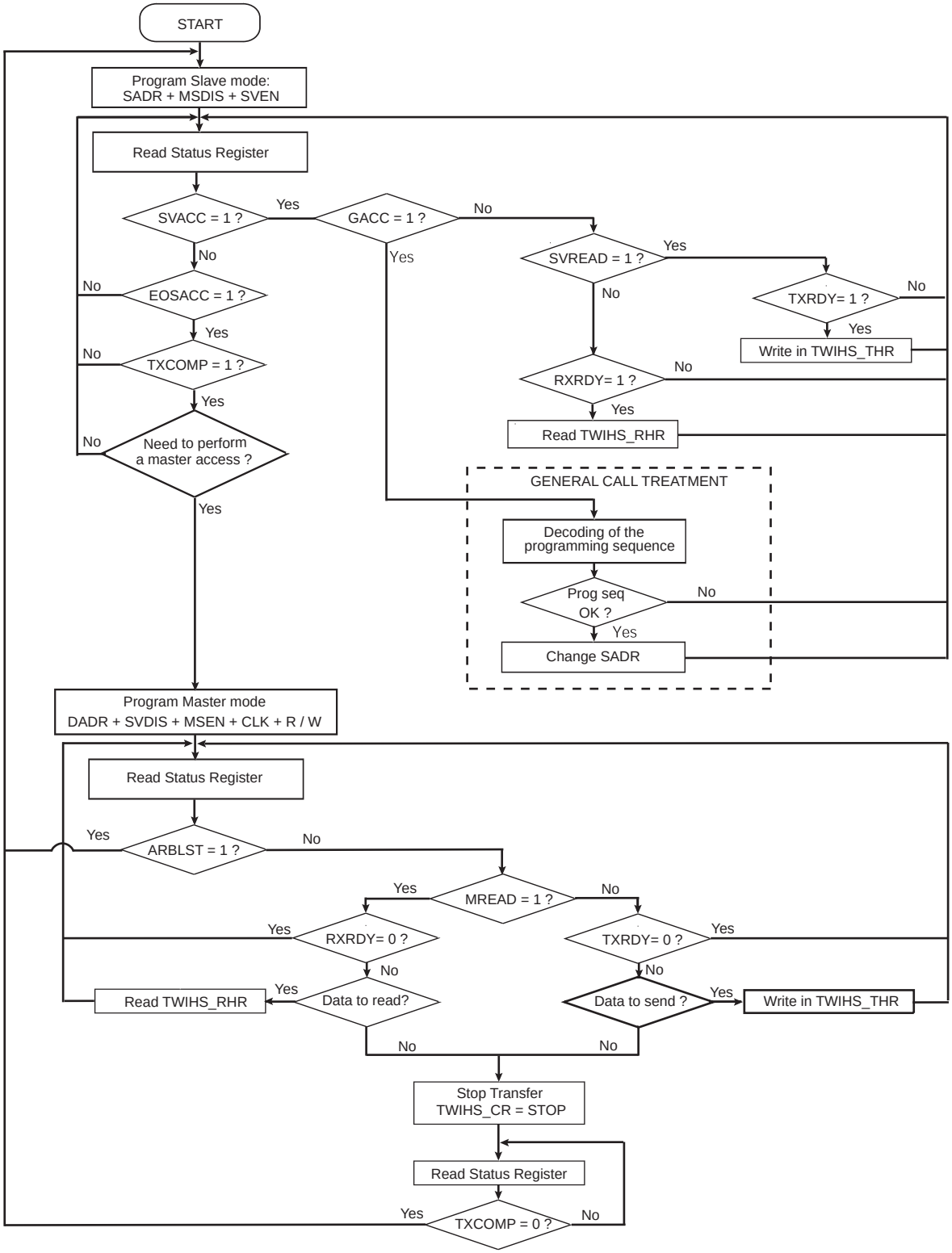


Figure 43-29. Arbitration Cases



The flowchart shown in Figure 43-30 gives an example of read and write operations in Multimaster mode.

Figure 43-30. Multimaster Flowchart



43.6.5 Slave Mode

43.6.5.1 Definition

Slave mode is defined as a mode where the device receives the clock and the address from another device called the master.

In this mode, the device never initiates and never completes the transmission (START, REPEATED_START and STOP conditions are always provided by the master).

43.6.5.2 Programming Slave Mode

The following fields must be programmed before entering Slave mode:

1. TWIHS_SMR.SADR: The slave device address is used in order to be accessed by master devices in Read or Write mode.
2. (Optional) TWIHS_SMR.MASK can be set to mask some SADR address bits and thus allow multiple address matching.
3. TWIHS_CR.MSDIS: Disables the Master mode.
4. TWIHS_CR.SVEN: Enables the Slave mode.

As the device receives the clock, values written in TWIHS_CWGR are ignored.

43.6.5.3 Receiving Data

After a START or REPEATED START condition is detected, and if the address sent by the master matches the slave address programmed in the SADR (Slave Address) field, the SVACC (Slave Access) flag is set and SVREAD (Slave Read) indicates the direction of the transfer.

SVACC remains high until a STOP condition or a REPEATED START is detected. When such a condition is detected, the EOSACC (End Of Slave Access) flag is set.

Read Sequence

In the case of a read sequence (SVREAD is high), the TWIHS transfers data written in the TWIHS_THR until a STOP condition or a REPEATED_START + an address different from SADR is detected. Note that at the end of the read sequence TXCOMP (Transmission Complete) flag is set and SVACC reset.

As soon as data is written in the TWIHS_THR, TXRDY (Transmit Holding Register Ready) flag is reset, and it is set when the internal shifter is empty and the sent data acknowledged or not. If the data is not acknowledged, the NACK flag is set.

Note that a STOP or a REPEATED START always follows a NACK.

To clear the TXRDY flag, first set the bit TWIHS_CR.SVDIS, then set the bit TWIHS_CR.SVEN.

See Figure 43-31.

Write Sequence

In the case of a write sequence (SVREAD is low), the RXRDY (Receive Holding Register Ready) flag is set as soon as a character has been received in the TWIHS_RHR. RXRDY is reset when reading the TWIHS_RHR.

The TWIHS continues receiving data until a STOP condition or a REPEATED_START + an address different from SADR is detected. Note that at the end of the write sequence, the TXCOMP flag is set and SVACC is reset.

See Figure 43-32.

Clock Stretching Sequence

If TWIHS_THR or TWIHS_RHR is not written/read in time, the TWIHS performs a clock stretching.

Clock stretching information is given by the SCLWS (Clock Wait State) bit.

See Figure 43-34 and Figure 43-35.

Note: Clock stretching can be disabled by configuring the SCLWSDIS bit in the TWIHS_SMR. In that case, the UNRE and OVRE flags indicate an underrun (when TWIHS_THR is not filled on time) or an overrun (when TWIHS_RHR is not read on time).

General Call

In the case where a GENERAL CALL is performed, the GACC (General Call Access) flag is set.

After GACC is set, the user must interpret the meaning of the GENERAL CALL and decode the new address programming sequence.

See Figure 43-33.

43.6.5.4 Data Transfer

Read Operation

The Read mode is defined as a data requirement from the master.

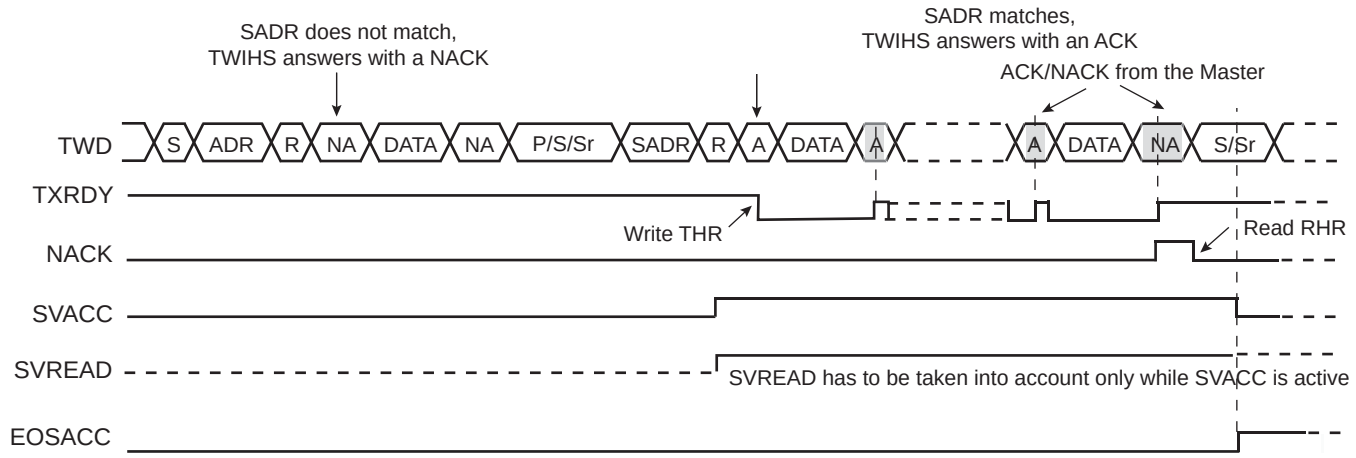
After a START or a REPEATED START condition is detected, the decoding of the address starts. If the slave address (SADR) is decoded, SVACC is set and SVREAD indicates the direction of the transfer.

Until a STOP or REPEATED START condition is detected, the TWIHS continues sending data loaded in the TWIHS_THR.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 43-31 describes the read operation.

Figure 43-31. Read Access Ordered by a Master



Notes:
1. When SVACC is low, the state of SVREAD becomes irrelevant.
2. TXRDY is reset when data has been transmitted from TWIHS_THR to the internal shifter and set when this data has been acknowledged or non acknowledged.

Write Operation

The Write mode is defined as a data transmission from the master.

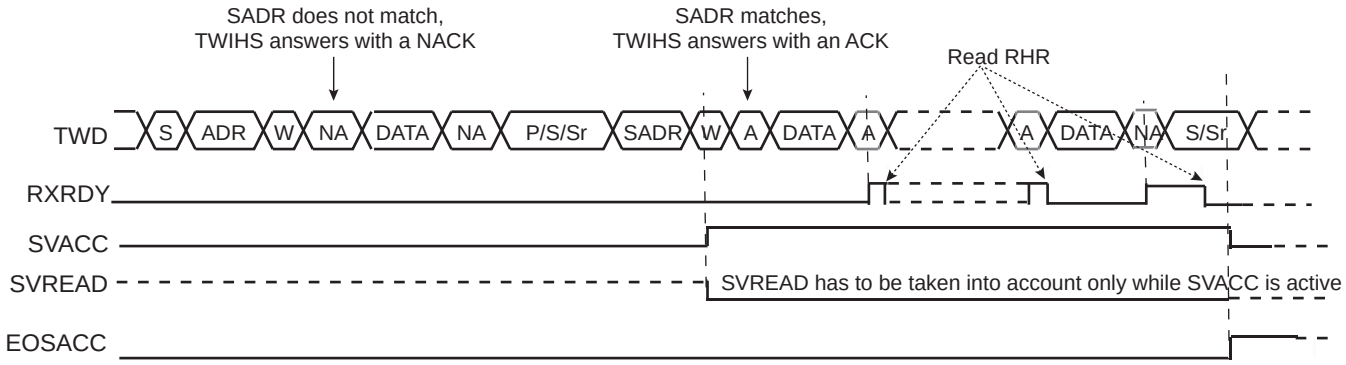
After a START or a REPEATED START, the decoding of the address starts. If the slave address is decoded, SVACC is set and SVREAD indicates the direction of the transfer (SVREAD is low in this case).

Until a STOP or REPEATED START condition is detected, the TWIHS stores the received data in the TWIHS_RHR.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 43-32 describes the write operation.

Figure 43-32. Write Access Ordered by a Master



Notes:
1. When SVACC is low, the state of SVREAD becomes irrelevant.
2. RXRDY is set when data has been transmitted from the internal shifter to the TWIHS_RHR and reset when this data is read.

General Call

The general call is performed in order to change the address of the slave.

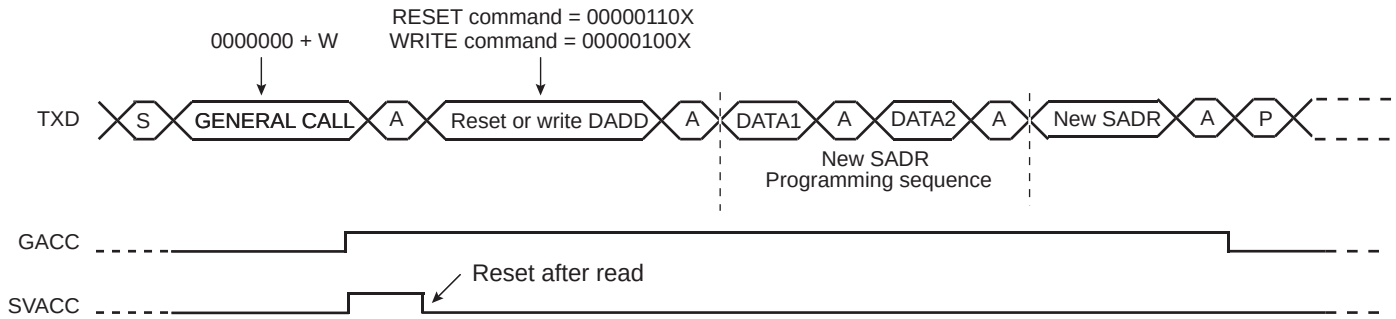
If a GENERAL CALL is detected, GACC is set.

After the detection of general call, decode the commands that follow.

In case of a WRITE command, decode the programming sequence and program a new SADR if the programming sequence matches.

Figure 43-33 describes the general call access.

Figure 43-33. Master Performs a General Call



Note: This method enables the user to create a personal programming sequence by choosing the programming bytes and their number. The programming sequence has to be provided to the master.

Clock Stretching

In both Read and Write modes, it may occur that TWIHS_THR/TWIHS_RHR buffer is not filled/emptied before the transmission/reception of a new character. In this case, to avoid sending/receiving undesired data, a clock stretching mechanism is implemented.

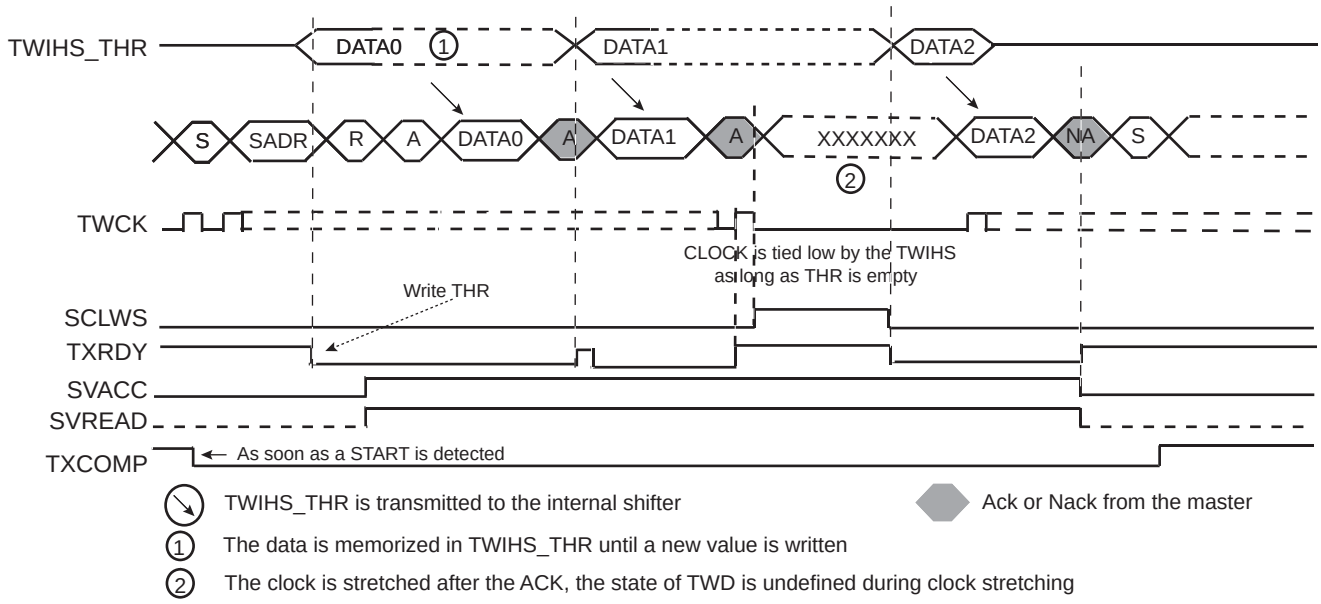
Note: Clock stretching can be disabled by setting the SCLWSDIS bit in the TWIHS_SMR. In that case the UNRE and OVRE flags indicate an underrun (when TWIHS_THR is not filled on time) or an overrun (when TWIHS_RHR is not read on time).

Clock Stretching in Read Mode

The clock is tied low if the internal shifter is empty and if a STOP or REPEATED START condition was not detected. It is tied low until the internal shifter is loaded.

Figure 43-34 describes the clock stretching in Read mode.

Figure 43-34. Clock Stretching in Read Mode



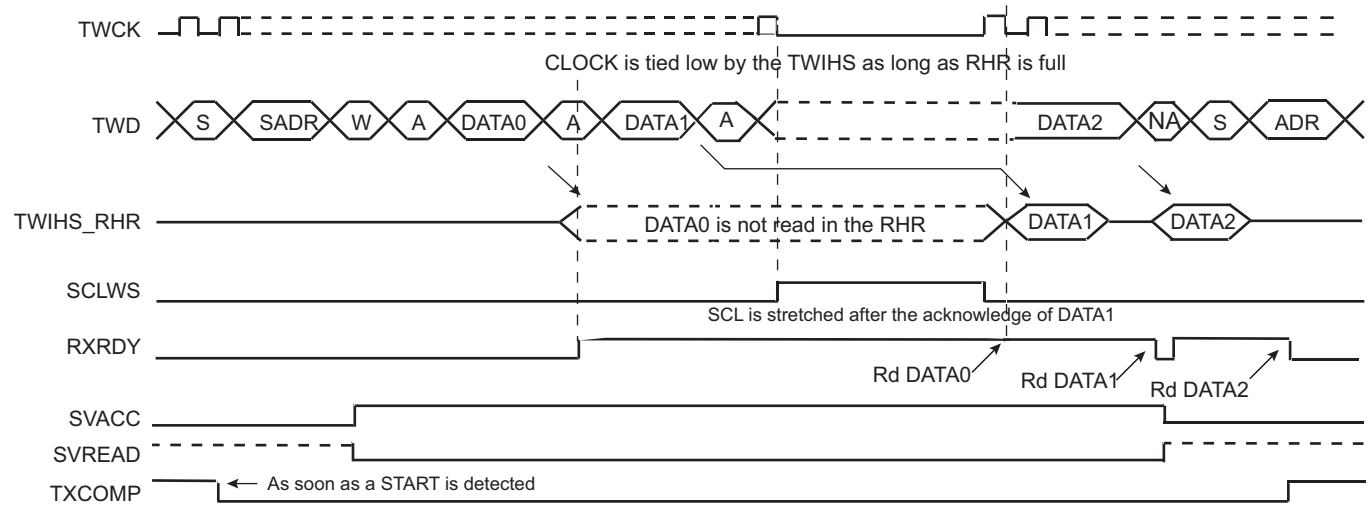
Notes:
1. TXRDY is reset when data has been written in the TWIHS_THR to the internal shifter and set when this data has been acknowledged or non acknowledged.
2. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.
3. SCLWS is automatically set when the clock stretching mechanism is started.

Clock Stretching in Write Mode

The clock is tied low if the internal shifter and the TWIHS_RHR is full. If a STOP or REPEATED_START condition was not detected, it is tied low until TWIHS_RHR is read.

Figure 43-35 describes the clock stretching in Write mode.

Figure 43-35. Clock Stretching in Write Mode



Notes:
1. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.
2. SCLWS is automatically set when the clock stretching mechanism is started and automatically reset when the mechanism is finished.

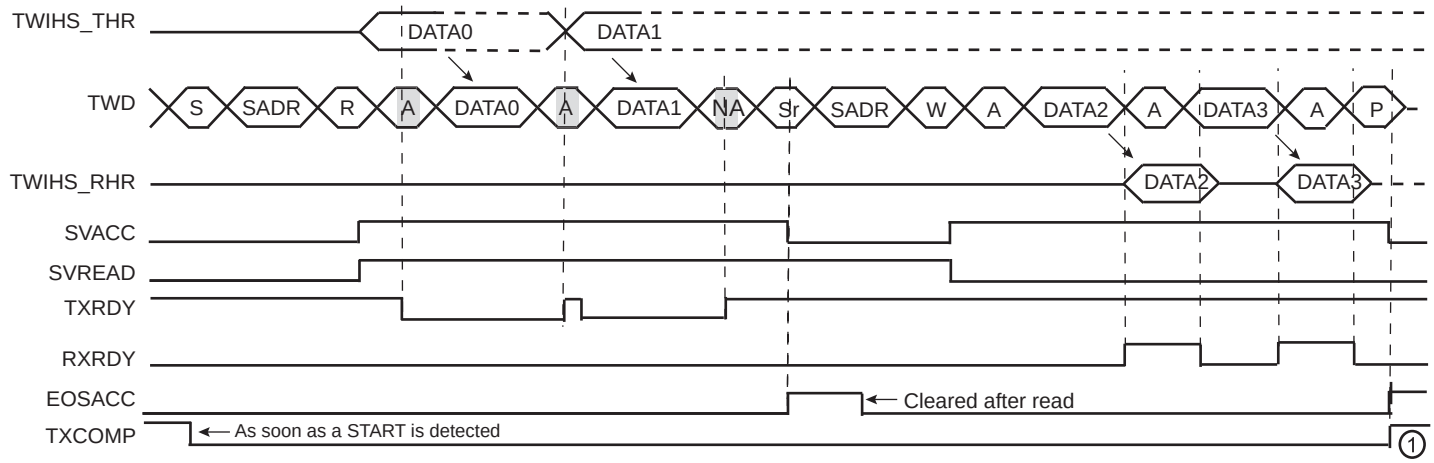
Reversal after a Repeated Start

Reversal of Read to Write

The master initiates the communication by a read command and finishes it by a write command.

Figure 43-36 describes the REPEATED START and the reversal from Read mode to Write mode.

Figure 43-36. Repeated Start and Reversal from Read Mode to Write Mode

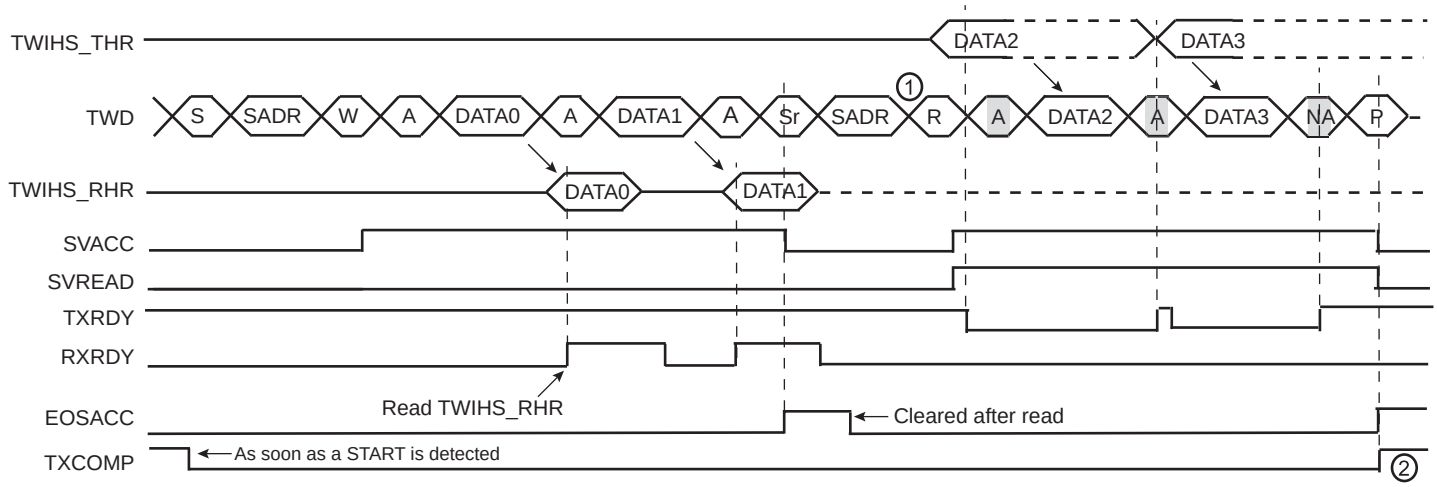


Note: TXCOMP is only set at the end of the transmission. This is because after the REPEATED START, SADR is detected again.

Reversal of Write to Read

The master initiates the communication by a write command and finishes it by a read command. Figure 43-37 describes the REPEATED START and the reversal from Write mode to Read mode.

Figure 43-37. Repeated Start and Reversal from Write Mode to Read Mode



Notes:
1. In this case, if TWIHS_THR has not been written at the end of the read command, the clock is automatically stretched before the ACK.
2. TXCOMP is only set at the end of the transmission. This is because after the REPEATED START, SADR is detected again.

43.6.5.5 Using the DMA Controller (DMAC) in Slave Mode

The use of the DMAC significantly reduces the CPU load.

Data Transmit with the DMA in Slave Mode

The following procedure shows an example to transmit data with DMA.

1. Initialize the transmit DMA (memory pointers, transfer size, etc).
2. Configure the Slave mode.
3. Enable the DMA.
4. Wait for the DMA status flag indicating that the buffer transfer is complete.
5. Disable the DMA.
6. (Only if peripheral clock must be disabled) Wait for the TXCOMP flag to be raised in TWIHS_SR.

Data Receive with the DMA in Slave Mode

The following procedure shows an example to transmit data with DMA where the number of characters to receive is known.

1. Initialize the DMA (channels, memory pointers, size, etc.).
2. Configure the Slave mode.
3. Enable the DMA.
4. Wait for the DMA status flag indicating that the buffer transfer is complete.
5. Disable the DMA.
6. (Only if peripheral clock must be disabled) Wait for the TXCOMP flag to be raised in TWIHS_SR.

43.6.5.6 SMBus Mode

SMBus mode is enabled when a one is written to the SMEN bit in the TWIHS_CR. SMBus mode operation is similar to I²C operation with the following exceptions:

- Only 7-bit addressing can be used.
- The SMBus standard describes a set of timeout values to ensure progress and throughput on the bus. These timeout values must be programmed into the TWIHS_SMBTR.
- Transmissions can optionally include a CRC byte, called Packet Error Check (PEC).
- A set of addresses have been reserved for protocol handling, such as alert response address (ARA) and host header (HH) address. Address matching on these addresses can be enabled by configuring the TWIHS_CR.

Packet Error Checking

Each SMBus transfer can optionally end with a CRC byte, called the PEC byte. Writing a one to the PECEN bit in TWIHS_CR will send/check the PEC field in the current transfer. The PEC generator is always updated on every bit transmitted or received, so that PEC handling on the following linked transfers is correct.

In Slave Receiver mode, the master calculates a PEC value and transmits it to the slave after all data bytes have been transmitted. Upon reception of this PEC byte, the slave compares it to the PEC value it has computed itself. If the values match, the data was received correctly, and the slave returns an ACK to the master. If the PEC values differ, data was corrupted, and the slave returns a NACK value. The PECERR bit in TWIHS_SR is set automatically if a PEC error occurred.

In Slave Transmitter mode, the slave calculates a PEC value and transmits it to the master after all data bytes have been transmitted. Upon reception of this PEC byte, the master compares it to the PEC value it has computed itself. If the values match, the data was received correctly. If the PEC values differ, data was corrupted, and the master must take appropriate action.

See Section 43.6.5.9 ”Slave Read Write Flowcharts” for detailed flowcharts.

Timeouts

The TWIHS SMBus Timing Register (TWIHS_SMBTR) configures the SMBus timeout values. If a timeout occurs, the slave leaves the bus. The TOUT bit is also set in TWIHS_SR.

43.6.5.7 High-Speed Slave Mode

High-speed mode is enabled when a one is written to the HSEN bit in the TWIHS_CR. Furthermore, the analog pad filter must be enabled, a one must be written to the PADFEN bit in the TWIHS_FILTR and the FILT bit must be cleared. TWIHS High-speed mode operation is similar to TWIHS operation with the following exceptions:

1. A master code is received first at normal speed before entering High-speed mode period.
2. When TWIHS High-speed mode is active, clock stretching is only allowed after acknowledge (ACK), not-acknowledge (NACK), START (S) or REPEATED START (Sr) (as consequence OVF may happen).

TWIHS High-speed mode allows transfers of up to 3.4 Mbit/s.

The TWIHS slave in High-speed mode requires that slave clock stretching is disabled (SCLWSDIS bit at ‘1’). The peripheral clock must run at a minimum of 11 MHz (assuming the system has no latency).

Note: When slave clock stretching is disabled, the TWIHS_RHR must always be read before receiving the next data (MASTER write frame). It is strongly recommended to use either the polling method on the RXRDY flag in TWIHS_SR, or the DMA. If the receive is managed by an interrupt, the TWIHS interrupt priority must be set to the right level and its latency minimized to avoid receive overrun.

Note: When slave clock stretching is disabled, the TWIHS_THR must be filled with the first data to send before the beginning of the frame (MASTER read frame). It is strongly recommended to use either the polling method on the TXRDY flag in TWIHS_SR, or the DMA. If the transmit is managed by an interrupt, the TWIHS interrupt priority must be set to the right level and its latency minimized to avoid transmit underrun.

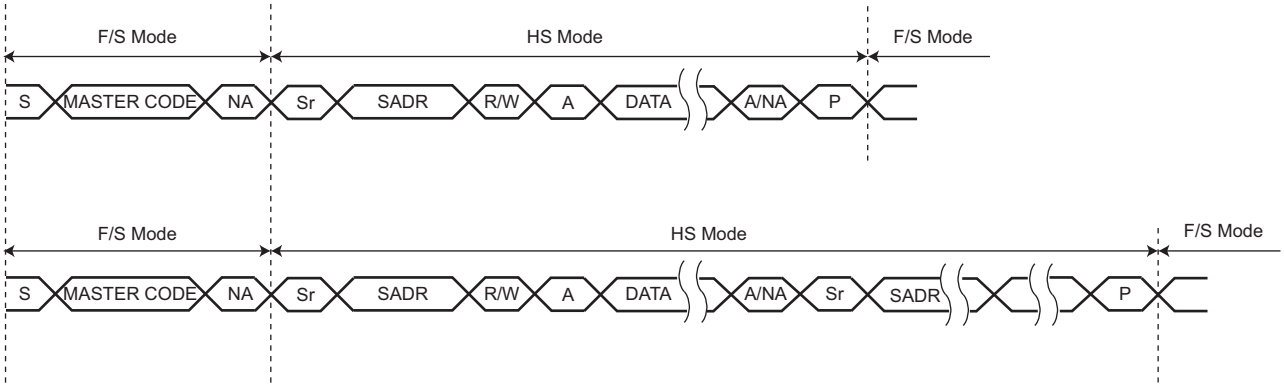
Read/Write Operation

A TWIHS high-speed frame always begins with the following sequence:

1. START condition (S)
2. Master Code (0000 1XXX)
3. Not-acknowledge (NACK)

When the TWIHS is programmed in Slave mode and TWIHS High-speed mode is activated, master code matching is activated and internal timings are set to match the TWIHS High-speed mode requirements.

Figure 43-38. High-Speed Mode Read/Write



Usage

TWIHS High-speed mode usage is the same as the standard TWIHS (See Section 43.6.3.11 ”Read/Write Flowcharts”).

43.6.5.8 Asynchronous Partial Wakeup (SleepWalking)

The TWIHS includes an asynchronous start condition detector. It is capable of waking the device up from a Sleep mode upon an address match (and optionally an additional data match), including Sleep modes where the TWIHS peripheral clock is stopped.

After detecting the START condition on the bus, the TWIHS stretches TWCK until the TWIHS peripheral clock has started. The time required for starting the TWIHS depends on which Sleep mode the device is in. After the TWIHS peripheral clock has started, the TWIHS releases its TWCK stretching and receives one byte of data (slave address) on the bus. At this time, only a limited part of the device, including the TWIHS module, receives a clock, thus saving power. If the address phase causes a TWIHS address match (and, optionally, if the first data byte causes data match as well), the entire device is woken up and normal TWIHS address matching actions are performed. Normal TWIHS transfer then follows. If the TWIHS is not addressed (or if the optional data match fails), the TWIHS peripheral clock is automatically stopped and the device returns to its original Sleep mode.

The TWIHS has the capability to match on more than one address. The SADR1EN, SADR2EN and SADR3EN bits in TWIHS_SMR enable address matching on additional addresses which can be configured through SADR1, SADR2 and SADR3 fields in the TWIHS_SWMR. The SleepWalking matching process can be extended to the first received data byte if DATAMEN bit in TWIHS_SMR is set and, in this case, a complete matching includes address matching and first received data matching. The field DATAM in TWIHS_SWMR configures the data to match on the first received byte.

When the system is in Active mode and the TWIHS enters Asynchronous Partial Wakeup mode, the flag SVACC must be programmed as the unique source of the TWIHS interrupt and the data match comparison must be disabled.

When the system exits Wait mode as the result of a matching condition, the SVACC flag is used to determine if the TWIHS is the source of exit.

Figure 43-39. Address Match Only (Data Matching Disabled)

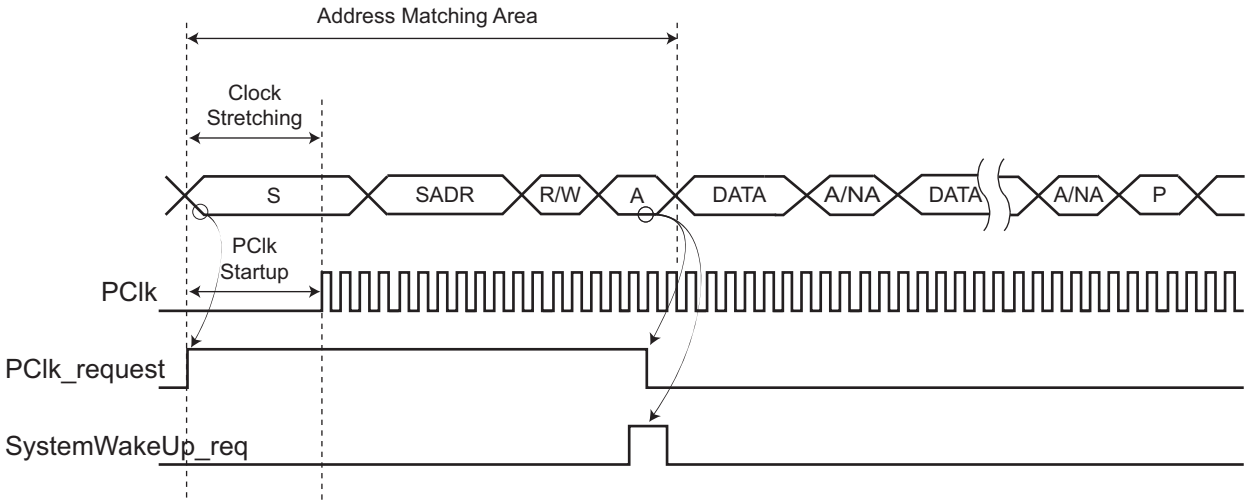


Figure 43-40. No Address Match (Data Matching Disabled)

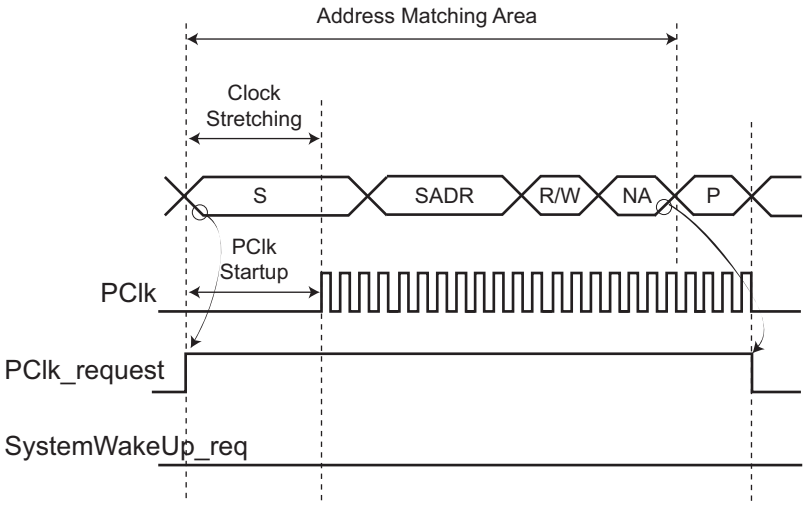


Figure 43-41. Address Match and Data Match (Data Matching Enabled)

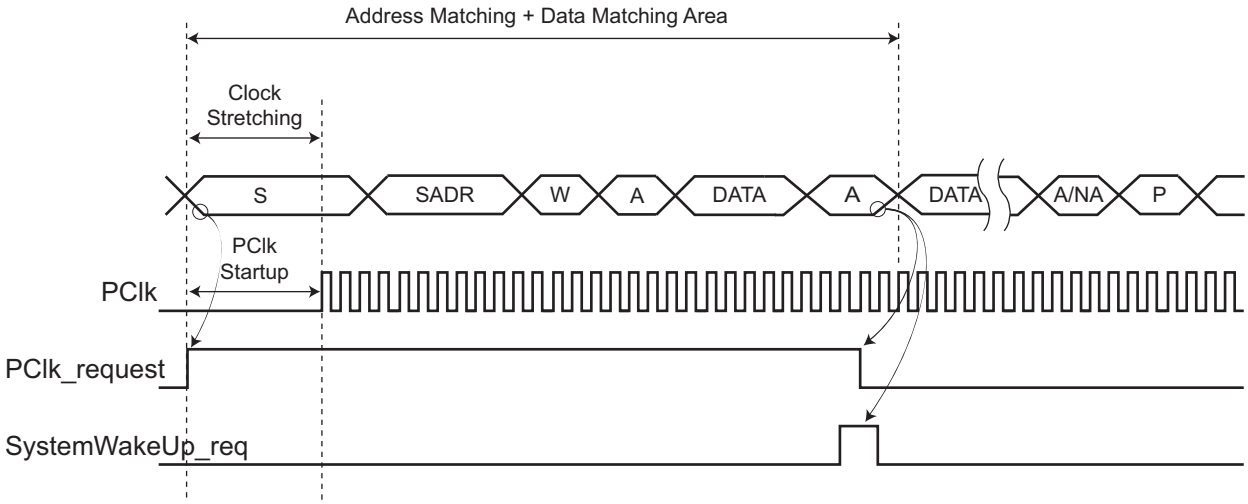
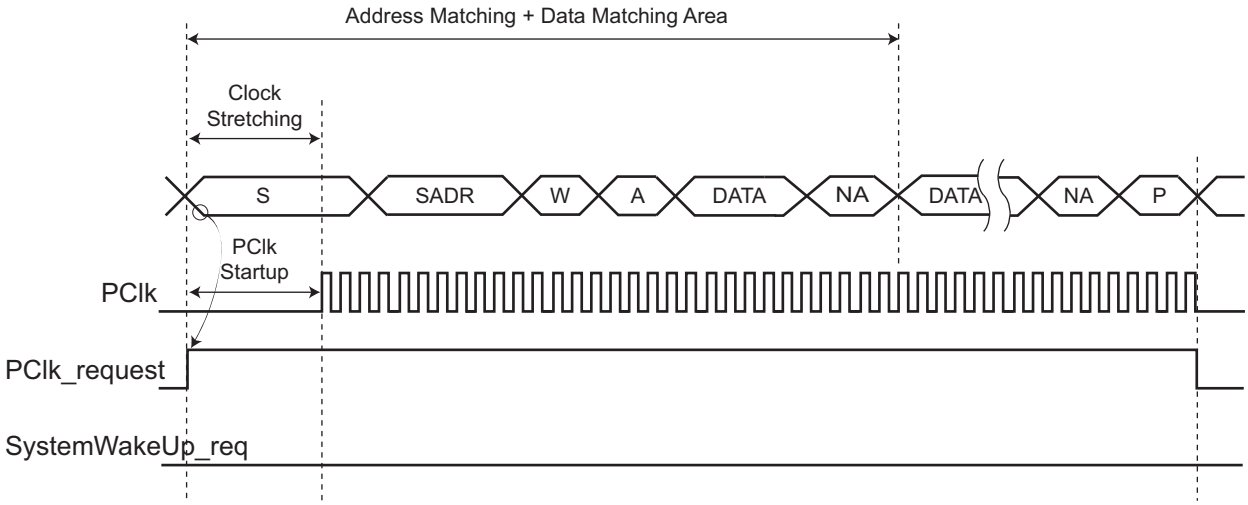


Figure 43-42. Address Match and No Data Match (Data Matching Enabled)



43.6.5.9 Slave Read Write Flowcharts

The flowchart shown in Figure 43-43 gives an example of read and write operations in Slave mode. A polling or interrupt method can be used to check the status bits. The interrupt method requires that TWIHS_IER be configured first.

Figure 43-43. Read Write Flowchart in Slave Mode

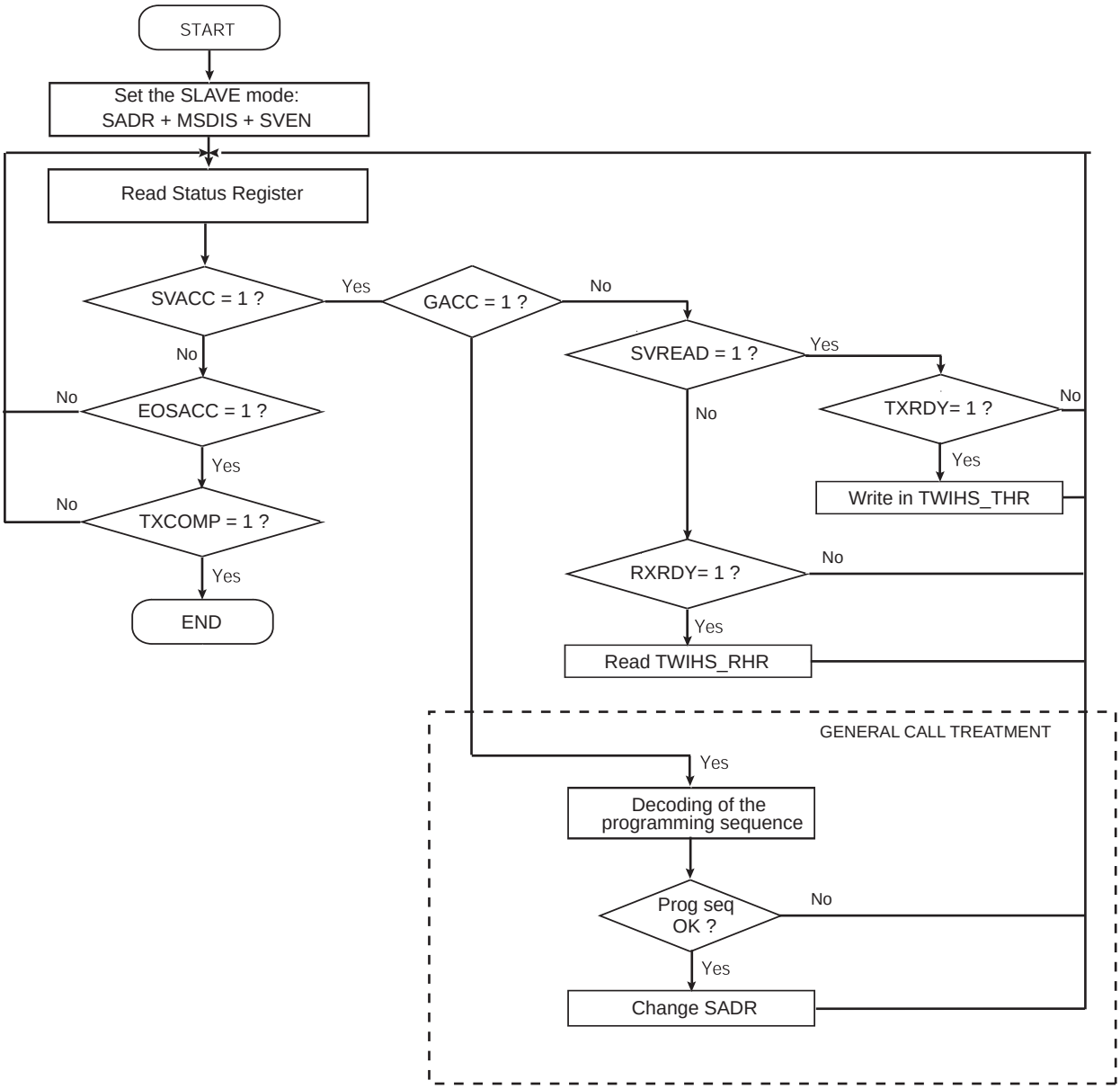


Figure 43-44. Read Write Flowchart in Slave Mode with SMBus PEC

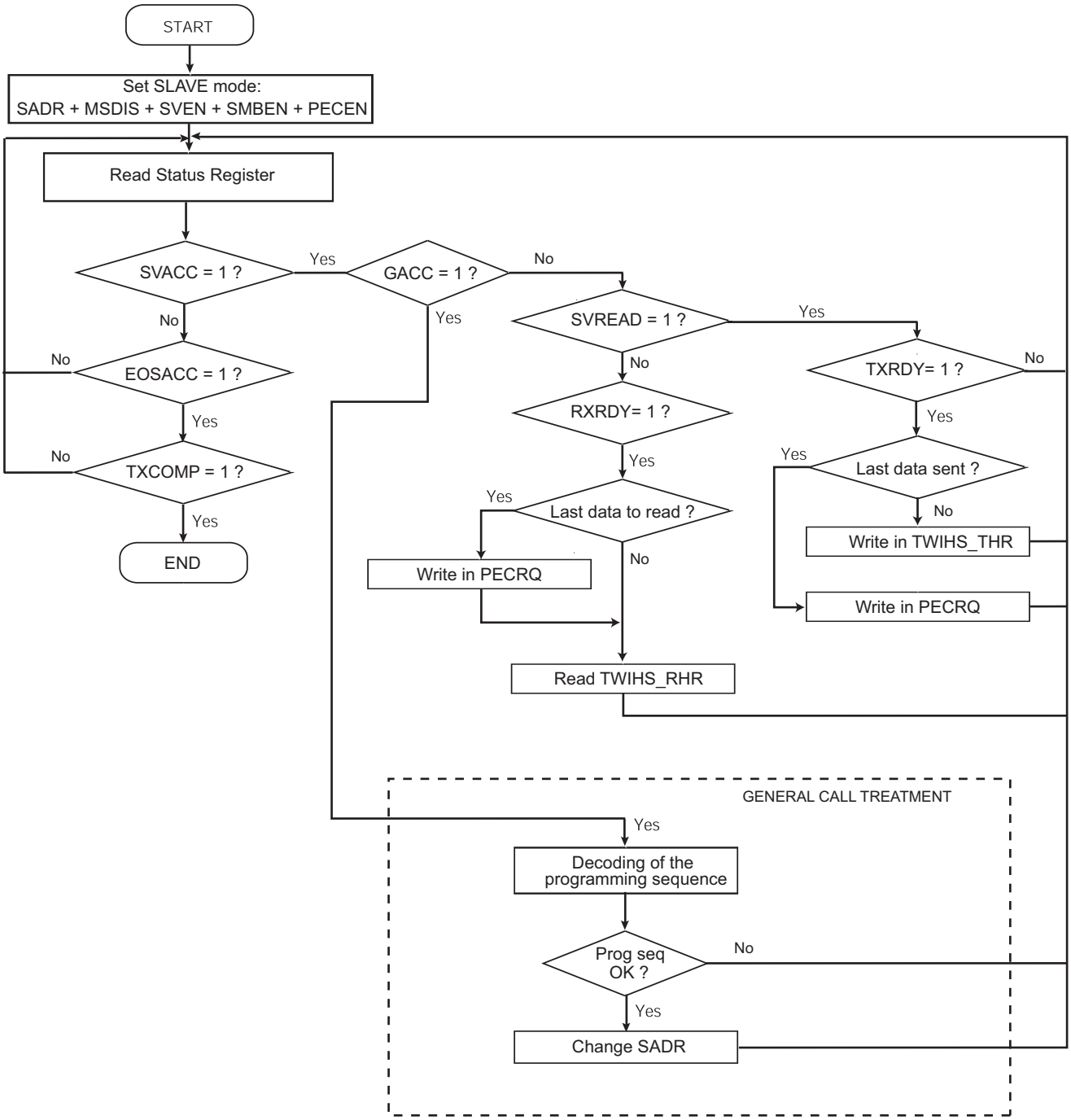
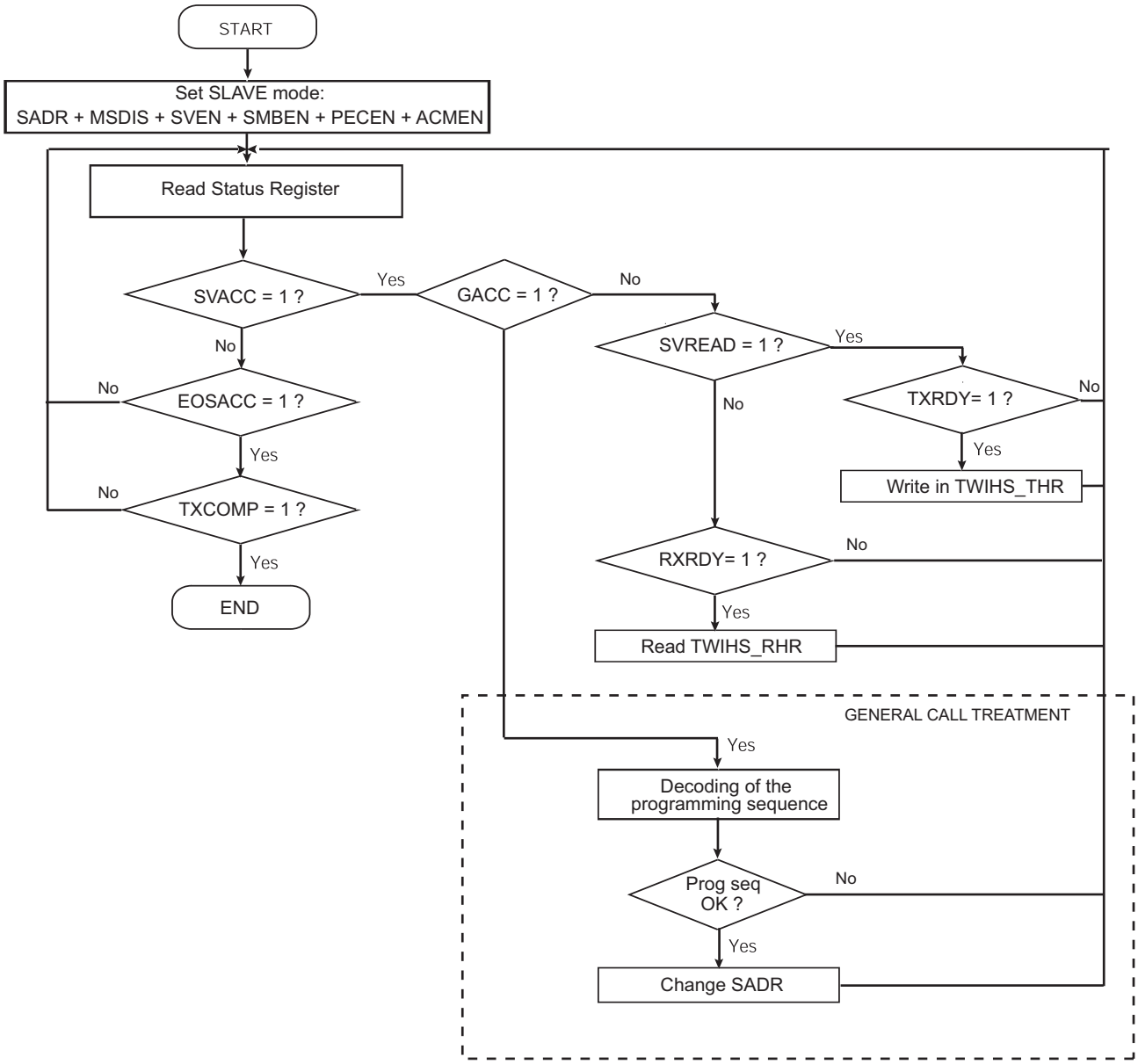


Figure 43-45. Read Write Flowchart in Slave Mode with SMBus PEC and Alternative Command Mode



43.6.6 TWIHS Comparison Function on Received Character

The comparison function differs if asynchronous partial wakeup (SleepWalking) is enabled or not.

If asynchronous partial wakeup is disabled (see the section “Power Management Controller (PMC)”), the TWIHS can extend the address matching on up to three slave addresses. The SADR1EN, SADR2EN and SADR3EN bits in TWIHS_SMR enable address matching on additional addresses which can be configured through SADR1, SADR2 and SADR3 fields in the TWIHS_SWMR. The DATAMEN bit in the TWIHS_SMR has no effect.

The SVACC bit is set when there is a comparison match with the received slave address.

43.6.7 Register Write Protection

To prevent any single software error from corrupting TWIHS behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the TWIHS Write Protection Mode Register (TWIHS_WPMR).

If a write access to a write-protected register is detected, the WPVS bit in the TWIHS Write Protection Status Register (TWIHS_WPSR) is set and the field WPVSRC indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading TWIHS_WPSR.

The following registers can be write-protected:

- TWIHS Clock Waveform Generator Register

43.7 Two-wire Interface High Speed (TWIHS) User Interface

Table 43-7. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	TWIHS_CR	Write-only	–
0x04	Master Mode Register	TWIHS_MMR	Read/Write	0x00000000
0x08	Slave Mode Register	TWIHS_SMR	Read/Write	0x00000000
0x0C	Internal Address Register	TWIHS_IADR	Read/Write	0x00000000
0x10	Clock Waveform Generator Register	TWIHS_CWGR	Read/Write	0x00000000
0x14–0x1C	Reserved	–	–	–
0x20	Status Register	TWIHS_SR	Read-only	0x0300F009
0x24	Interrupt Enable Register	TWIHS_IER	Write-only	–
0x28	Interrupt Disable Register	TWIHS_IDR	Write-only	–
0x2C	Interrupt Mask Register	TWIHS_IMR	Read-only	0x00000000
0x30	Receive Holding Register	TWIHS_RHR	Read-only	0x00000000
0x34	Transmit Holding Register	TWIHS_THR	Write-only	0x00000000
0x38	SMBus Timing Register	TWIHS_SMBTR	Read/Write	0x00000000
0x3C	Reserved	–	–	–
0x40	Reserved	–	–	–
0x44	Filter Register	TWIHS_FILTR	Read/Write	0x00000000
0x48	Reserved	–	–	–
0x4C	SleepWalking Matching Register	TWIHS_SWMR	Read/Write	0x00000000
0x50–0xCC	Reserved	–	–	–
0xD0	Reserved	–	–	–
0xD4–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	TWIHS_WPMR	Read/Write	0x00000000
0xE8	Write Protection Status Register	TWIHS_WPSR	Read-only	0x00000000
0xEC–0xFC[1]	Reserved	–	–	–
0x100–0x128	Reserved	–	–	–

Note: 1. All unlisted offset values are considered as “reserved”.

43.7.1 TWIHS Control Register

Name: TWIHS_CR

Address: 0x40018000 (0), 0x4001C000 (1), 0x40060000 (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	FIFODIS	FIFOEN	–	LOCKCLR	–	THRCLR
23	22	21	20	19	18	17	16
–	–	–	–	–	–	ACMDIS	ACMEN
15	14	13	12	11	10	9	8
CLEAR	PECRQ	PECDIS	PECEN	SMBDIS	SMBEN	HSDIS	HSEN
7	6	5	4	3	2	1	0
SWRST	QUICK	SVDIS	SVEN	MSDIS	MSEN	STOP	START

- **START: Send a START Condition**

0: No effect.

1: A frame beginning with a START bit is transmitted according to the features defined in the TWIHS Master Mode Register (TWIHS_MMR).

This action is necessary when the TWIHS peripheral needs to read data from a slave. When configured in Master mode with a write operation, a frame is sent as soon as the user writes a character in the Transmit Holding Register (TWIHS_THR).

- **STOP: Send a STOP Condition**

0: No effect.

1: STOP condition is sent just after completing the current byte transmission in Master Read mode.

  – In single data byte master read, both START and STOP must be set.
  – In multiple data bytes master read, the STOP must be set after the last data received but one.
  – In Master Read mode, if a NACK bit is received, the STOP is automatically performed.
  – In master data write operation, a STOP condition will be sent after the transmission of the current data is finished.

- **MSEN: TWIHS Master Mode Enabled**

0: No effect.

1: Enables the Master mode (MSDIS must be written to 0).

Note: Switching from Slave to Master mode is only permitted when TXCOMP = 1.

- **MSDIS: TWIHS Master Mode Disabled**

0: No effect.

1: The Master mode is disabled, all pending data is transmitted. The shifter and holding characters (if it contains data) are transmitted in case of write operation. In read operation, the character being transferred must be completely received before disabling.

- **SVEN: TWIHS Slave Mode Enabled**

0: No effect.

1: Enables the Slave mode (SVDIS must be written to 0).

Note: Switching from Master to Slave mode is only permitted when TXCOMP = 1.

- **SVDIS: TWIHS Slave Mode Disabled**

0: No effect.

1: The Slave mode is disabled. The shifter and holding characters (if it contains data) are transmitted in case of read operation. In write operation, the character being transferred must be completely received before disabling.

- **QUICK: SMBus Quick Command**

0: No effect.

1: If Master mode is enabled, a SMBus Quick Command is sent.

- **SWRST: Software Reset**

0: No effect.

1: Equivalent to a system reset.

- **HSEN: TWIHS High-Speed Mode Enabled**

0: No effect.

1: High-speed mode enabled.

- **HSDIS: TWIHS High-Speed Mode Disabled**

0: No effect.

1: High-speed mode disabled.

- **SMBEN: SMBus Mode Enabled**

0: No effect.

1: If SMBDIS = 0, SMBus mode enabled.

- **SMBDIS: SMBus Mode Disabled**

0: No effect.

1: SMBus mode disabled.

- **PECEN: Packet Error Checking Enable**

0: No effect.

1: SMBus PEC (CRC) generation and check enabled.

- **PECDIS: Packet Error Checking Disable**

0: No effect.

1: SMBus PEC (CRC) generation and check disabled.

- **PECRQ: PEC Request**

0: No effect.

1: A PEC check or transmission is requested.

- **CLEAR: Bus CLEAR Command**

0: No effect.

1: If Master mode is enabled, sends a bus clear command.

- **ACMEN: Alternative Command Mode Enable**

0: No effect.

1: Alternative Command mode enabled.

- **ACMDIS: Alternative Command Mode Disable**

0: No effect.

1: Alternative Command mode disabled.

- **THRCLR: Transmit Holding Register Clear**

0: No effect.

1: Clears the Transmit Holding Register and set TXRDY, TXCOMP flags.

- **LOCKCLR: Lock Clear**

0: No effect.

1: Clears the TWIHS FSM lock.

- **FIFOEN: FIFO Enable**

0: No effect.

1: Enables the Transmit and Receive FIFOs

- **FIFODIS: FIFO Disable**

0: No effect.

1: Disables the Transmit and Receive FIFOs

43.7.2 TWIHS Master Mode Register

Name: TWIHS_MMR

Address: 0x40018004 (0), 0x4001C004 (1), 0x40060004 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	DADR						

15	14	13	12	11	10	9	8
–	–	–	MREAD	–	–	IADRSZ	

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **IADRSZ: Internal Device Address Size**

Value	Name	Description
0	NONE	No internal device address
1	1_BYTE	One-byte internal device address
2	2_BYTE	Two-byte internal device address
3	3_BYTE	Three-byte internal device address

- **MREAD: Master Read Direction**

0: Master write direction.

1: Master read direction.

- **DADR: Device Address**

The device address is used to access slave devices in Read or Write mode. These bits are only used in Master mode.

43.7.3 TWIHS Slave Mode Register

Name: TWIHS_SMR

Address: 0x40018008 (0), 0x4001C008 (1), 0x40060008 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
DATAMEN	SADR3EN	SADR2EN	SADR1EN	–	–	–	–
23	22	21	20	19	18	17	16
–	SADR						
15	14	13	12	11	10	9	8
–	MASK						
7	6	5	4	3	2	1	0
–	SCLWSDIS	–	–	SMHH	SMDA	–	NACKEN

- **NACKEN: Slave Receiver Data Phase NACK enable**

0: Normal value to be returned in the ACK cycle of the data phase in Slave Receiver mode.

1: NACK value to be returned in the ACK cycle of the data phase in Slave Receiver mode.

- **SMDA: SMBus Default Address**

0: Acknowledge of the SMBus default address disabled.

1: Acknowledge of the SMBus default address enabled.

- **SMHH: SMBus Host Header**

0: Acknowledge of the SMBus host header disabled.

1: Acknowledge of the SMBus host header enabled.

- **SCLWSDIS: Clock Wait State Disable**

0: No effect.

1: Clock stretching disabled in Slave mode, OVRE and UNRE indicate an overrun/underrun.

- **MASK: Slave Address Mask**

A mask can be applied on the slave device address in Slave mode in order to allow multiple address answer. For each bit of the MASK field set to 1, the corresponding SADR bit is masked.

If the MASK field value is 0, no mask is applied to the SADR field.

- **SADR: Slave Address**

The slave device address is used in Slave mode in order to be accessed by master devices in Read or Write mode.

SADR must be programmed before enabling the Slave mode or after a general call. Writes at other times have no effect.

- **SADR1EN: Slave Address 1 Enable**

0: Slave address 1 matching is disabled.

1: Slave address 1 matching is enabled.

- **SADR2EN: Slave Address 2 Enable**

0: Slave address 2 matching is disabled.

1: Slave address 2 matching is enabled.

- **SADR3EN: Slave Address 3 Enable**

0: Slave address 3 matching is disabled.

1: Slave address 3 matching is enabled.

- **DATAMEN: Data Matching Enable**

0: Data matching on first received data is disabled.

1: Data matching on first received data is enabled.

43.7.4 TWIHS Internal Address Register

Name: TWIHS_IADR

Address: 0x4001800C (0), 0x4001C00C (1), 0x4006000C (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
IADR							

15	14	13	12	11	10	9	8
IADR							

7	6	5	4	3	2	1	0
IADR							

- **IADR: Internal Address**

0, 1, 2 or 3 bytes depending on IADRSZ.

43.7.5 TWIHS Clock Waveform Generator Register

Name: TWIHS_CWGR

Address: 0x40018010 (0), 0x4001C010 (1), 0x40060010 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	HOLD					

23	22	21	20	19	18	17	16
–	–	–	–	–	CKDIV		

15	14	13	12	11	10	9	8
CHDIV							

7	6	5	4	3	2	1	0
CLDIV							

This register can only be written if the WPEN bit is cleared in the TWIHS Write Protection Mode Register.

TWIHS_CWGR is used in Master mode only.

- **CLDIV: Clock Low Divider**

The SCL low period is defined as follows:

$t_{low} = ((CLDIV \times 2^{CKDIV}) + 3) \times t_{peripheral\ clock}$

- **CHDIV: Clock High Divider**

The SCL high period is defined as follows:

$t_{high} = ((CHDIV \times 2^{CKDIV}) + 3) \times t_{peripheral\ clock}$

- **CKDIV: Clock Divider**

The CKDIV is used to increase both SCL high and low periods.

- **HOLD: TWD Hold Time Versus TWCK Falling**

If High-speed mode is selected TWD is internally modified on the TWCK falling edge to meet the I2C specified maximum hold time, else if High-speed mode is not configured TWD is kept unchanged after TWCK falling edge for a period of $(HOLD + 3) \times t_{peripheral\ clock}$.

43.7.6 TWIHS Status Register

Name: TWIHS_SR

Address: 0x40018020 (0), 0x4001C020 (1), 0x40060020 (2)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	SDA	SCL
23	22	21	20	19	18	17	16
–	–	SMBHHM	SMBDAM	PECERR	TOUT	–	MCACK
15	14	13	12	11	10	9	8
–	–	–	–	EOSACC	SCLWS	ARBLST	NACK
7	6	5	4	3	2	1	0
UNRE	OVRE	GACC	SVACC	SVREAD	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed (cleared by writing TWIHS_THR)**

TXCOMP used in Master mode:

0: During the length of the current frame.

1: When both holding register and internal shifter are empty and STOP condition has been sent.

TXCOMP behavior in Master mode can be seen in Figure 43-6 and in Figure 43-8.

TXCOMP used in Slave mode:

0: As soon as a START is detected.

1: After a STOP or a REPEATED START + an address different from SADR is detected.

TXCOMP behavior in Slave mode can be seen in Figure 43-34, Figure 43-35, Figure 43-36 and Figure 43-37.

- **RXRDY: Receive Holding Register Ready (cleared by reading TWIHS_RHR)**

0: No character has been received since the last TWIHS_RHR read operation.

1: A byte has been received in the TWIHS_RHR since the last read.

RXRDY behavior in Master mode can be seen in Figure 43-7, Figure 43-8 and Figure 43-9.

RXRDY behavior in Slave mode can be seen in Figure 43-32, Figure 43-35, Figure 43-36 and Figure 43-37.

- **TXRDY: Transmit Holding Register Ready (cleared by writing TWIHS_THR)**

TXRDY used in Master mode:

0: The transmit holding register has not been transferred into the internal shifter. Set to 0 when writing into TWIHS_THR.

1: As soon as a data byte is transferred from TWIHS_THR to internal shifter or if a NACK error is detected, TXRDY is set at the same time as TXCOMP and NACK. TXRDY is also set when MSEN is set (enables TWIHS).

TXRDY behavior in Master mode can be seen in Figure 43-4, Figure 43-5 and Figure 43-6.

TXRDY used in Slave mode:

0: As soon as data is written in the TWIHS_THR, until this data has been transmitted and acknowledged (ACK or NACK).

1: Indicates that the TWIHS_THR is empty and that data has been transmitted and acknowledged.

If TXRDY is high and if a NACK has been detected, the transmission is stopped. Thus when TRDY = NACK = 1, the user must not fill TWIHS_THR to avoid losing it.

TXRDY behavior in Slave mode can be seen in Figure 43-31, Figure 43-34, Figure 43-36 and Figure 43-37.

- **SVREAD: Slave Read**

This bit is used in Slave mode only. When SVACC is low (no slave access has been detected) SVREAD is irrelevant.

0: Indicates that a write access is performed by a master.

1: Indicates that a read access is performed by a master.

SVREAD behavior can be seen in Figure 43-31, Figure 43-32, Figure 43-36 and Figure 43-37.

- **SVACC: Slave Access**

This bit is used in Slave mode only.

0: TWIHS is not addressed. SVACC is automatically cleared after a NACK or a STOP condition is detected.

1: Indicates that the address decoding sequence has matched (A master has sent SADR). SVACC remains high until a NACK or a STOP condition is detected.

SVACC behavior can be seen in Figure 43-31, Figure 43-32, Figure 43-36 and Figure 43-37.

- **GACC: General Call Access (cleared on read)**

This bit is used in Slave mode only.

0: No general call has been detected.

1: A general call has been detected. After the detection of general call, if need be, the user may acknowledge this access and decode the following bytes and respond according to the value of the bytes.

GACC behavior can be seen in Figure 43-33.

- **OVRE: Overrun Error (cleared on read)**

This bit is used only if clock stretching is disabled.

0: TWIHS_RHR has not been loaded while RXRDY was set.

1: TWIHS_RHR has been loaded while RXRDY was set. Reset by read in TWIHS_SR when TXCOMP is set.

- **UNRE: Underrun Error (cleared on read)**

This bit is used only if clock stretching is disabled.

0: TWIHS_THR has been filled on time.

1: TWIHS_THR has not been filled on time.

- **NACK: Not Acknowledged (cleared on read)**

NACK used in Master mode:

0: Each data byte has been correctly received by the far-end side TWIHS slave component.

1: A data or address byte has not been acknowledged by the slave component. Set at the same time as TXCOMP.

NACK used in Slave Read mode:

0: Each data byte has been correctly received by the master.

1: In Read mode, a data byte has not been acknowledged by the master. When NACK is set, the user must not fill TWIHS_THR even if TXRDY is set, because it means that the master stops the data transfer or re-initiate it.

Note: in Slave Write mode, all data are acknowledged by the TWIHS.

- **ARBLST: Arbitration Lost (cleared on read)**

This bit is used in Master mode only.

0: Arbitration won.

1: Arbitration lost. Another master of the TWIHS bus has won the multimaster arbitration. TXCOMP is set at the same time.

- **SCLWS: Clock Wait State**

This bit is used in Slave mode only.

0: The clock is not stretched.

1: The clock is stretched. TWIHS_THR / TWIHS_RHR buffer is not filled / emptied before the transmission / reception of a new character.

SCLWS behavior can be seen in Figure 43-34 and Figure 43-35.

- **EOSACC: End Of Slave Access (cleared on read)**

This bit is used in Slave mode only.

0: A slave access is being performing.

1: The Slave Access is finished. End Of Slave Access is automatically set as soon as SVACC is reset.

EOSACC behavior can be seen in Figure 43-36 and Figure 43-37.

- **MCACK: Master Code Acknowledge (cleared on read)**

MACK used in Slave mode:

0: No Master Code has been received since the last read of TWIHS_SR.

1: A Master Code has been received since the last read of TWIHS_SR.

- **TOUT: Timeout Error (cleared on read)**

0: No SMBus timeout occurred since the last read of TWIHS_SR.

1: An SMBus timeout occurred since the last read of TWIHS_SR.

- **PECERR: PEC Error (cleared on read)**

0: No SMBus PEC error occurred since the last read of TWIHS_SR.

1: An SMBus PEC error occurred since the last read of TWIHS_SR.

- **SMBDAM: SMBus Default Address Match (cleared on read)**

0: No SMBus Default Address received since the last read of TWIHS_SR.

1: An SMBus Default Address was received since the last read of TWIHS_SR.

- **SMBHHM: SMBus Host Header Address Match (cleared on read)**

0: No SMBus Host Header Address received since the last read of TWIHS_SR.

1: An SMBus Host Header Address was received since the last read of TWIHS_SR.

- **SCL: SCL Line Value**

0: SCL line sampled value is ‘0’.

1: SCL line sampled value is ‘1.’

- **SDA: SDA Line Value**

0: SDA line sampled value is ‘0’.

1: SDA line sampled value is ‘1’.

43.7.7 TWIHS SMBus Timing Register

Name: TWIHS_SMBTR

Address: 0x40018038 (0), 0x4001C038 (1), 0x40060038 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
THMAX							
23	22	21	20	19	18	17	16
TLOWM							
15	14	13	12	11	10	9	8
TLOWS							
7	6	5	4	3	2	1	0
–	–	–	–	PRESC			

- **PRESC: SMBus Clock Prescaler**

Used to specify how to prescale the TLOWS, TLOWM and THMAX counters in SMBTR. Counters are prescaled according to the following formula:

$$f_{Prescaled} = \frac{f_{peripheral\ clock}}{2^{(PRESC+1)}}$$

- **TLOWS: Slave Clock Stretch Maximum Cycles**

0: TLOW:SEXT timeout check disabled.

1–255: Clock cycles in slave maximum clock stretch count. Prescaled by PRESC. Used to time TLOW:SEXT.

- **TLOWM: Master Clock Stretch Maximum Cycles**

0: TLOW:MEXT timeout check disabled.

1–255: Clock cycles in master maximum clock stretch count. Prescaled by PRESC. Used to time TLOW:MEXT.

- **THMAX: Clock High Maximum Cycles**

Clock cycles in clock high maximum count. Prescaled by PRESC. Used for bus free detection. Used to time THIGH:MAX.

43.7.8 TWIHS Filter Register

Name: TWIHS_FILTR

Address: 0x40018044 (0), 0x4001C044 (1), 0x40060044 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	THRES		
7	6	5	4	3	2	1	0
–	–	–	–	–	PADFCFG	PADFEN	FILT

- **FILT: RX Digital Filter**

0: No filtering applied on TWIHS inputs.

1: TWIHS input filtering is active (only in Standard and Fast modes)

Note: TWIHS digital input filtering follows a majority decision based on three samples from SDA/SCL lines at peripheral clock frequency.

- **PADFEN: PAD Filter Enable**

0: PAD analog filter is disabled.

1: PAD analog filter is enabled. (The analog filter must be enabled if High-speed mode is enabled.)

- **PADFCFG: PAD Filter Config**

See the electrical characteristics section for filter configuration details.

- **THRES: Digital Filter Threshold**

0: No filtering applied on TWIHS inputs.

1–7: Maximum pulse width of spikes to be suppressed by the input filter, defined in peripheral clock cycles.

43.7.9 TWIHS Interrupt Enable Register

Name: TWIHS_IER

Address: 0x40018024 (0), 0x4001C024 (1), 0x40060024 (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	SMBHHM	SMBDAM	PECERR	TOUT	–	MCACK
15	14	13	12	11	10	9	8
–	–	–	–	EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
UNRE	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **TXCOMP: Transmission Completed Interrupt Enable**
- **RXRDY: Receive Holding Register Ready Interrupt Enable**
- **TXRDY: Transmit Holding Register Ready Interrupt Enable**
- **SVACC: Slave Access Interrupt Enable**
- **GACC: General Call Access Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **UNRE: Underrun Error Interrupt Enable**
- **NACK: Not Acknowledge Interrupt Enable**
- **ARBLST: Arbitration Lost Interrupt Enable**
- **SCL_WS: Clock Wait State Interrupt Enable**
- **EOSACC: End Of Slave Access Interrupt Enable**
- **MCACK: Master Code Acknowledge Interrupt Enable**
- **TOUT: Timeout Error Interrupt Enable**
- **PECERR: PEC Error Interrupt Enable**
- **SMBDAM: SMBus Default Address Match Interrupt Enable**
- **SMBHHM: SMBus Host Header Address Match Interrupt Enable**

43.7.10 TWIHS Interrupt Disable Register

Name: TWIHS_IDR

Address: 0x40018028 (0), 0x4001C028 (1), 0x40060028 (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	SMBHHM	SMBDAM	PECERR	TOUT	–	MCACK
15	14	13	12	11	10	9	8
–	–	–	–	EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
UNRE	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

- **TXCOMP: Transmission Completed Interrupt Disable**
- **RXRDY: Receive Holding Register Ready Interrupt Disable**
- **TXRDY: Transmit Holding Register Ready Interrupt Disable**
- **SVACC: Slave Access Interrupt Disable**
- **GACC: General Call Access Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **UNRE: Underrun Error Interrupt Disable**
- **NACK: Not Acknowledge Interrupt Disable**
- **ARBLST: Arbitration Lost Interrupt Disable**
- **SCL_WS: Clock Wait State Interrupt Disable**
- **EOSACC: End Of Slave Access Interrupt Disable**
- **MCACK: Master Code Acknowledge Interrupt Disable**
- **TOUT: Timeout Error Interrupt Disable**
- **PECERR: PEC Error Interrupt Disable**
- **SMBDAM: SMBus Default Address Match Interrupt Disable**
- **SMBHHM: SMBus Host Header Address Match Interrupt Disable**

43.7.11 TWIHS Interrupt Mask Register

Name: TWIHS_IMR

Address: 0x4001802C (0), 0x4001C02C (1), 0x4006002C (2)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	SMBHHM	SMBDAM	PECERR	TOUT	–	MCACK
15	14	13	12	11	10	9	8
–	–	–	–	EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
UNRE	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

- **TXCOMP: Transmission Completed Interrupt Mask**
- **RXRDY: Receive Holding Register Ready Interrupt Mask**
- **TXRDY: Transmit Holding Register Ready Interrupt Mask**
- **SVACC: Slave Access Interrupt Mask**
- **GACC: General Call Access Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **UNRE: Underrun Error Interrupt Mask**
- **NACK: Not Acknowledge Interrupt Mask**
- **ARBLST: Arbitration Lost Interrupt Mask**
- **SCL_WS: Clock Wait State Interrupt Mask**
- **EOSACC: End Of Slave Access Interrupt Mask**
- **MCACK: Master Code Acknowledge Interrupt Mask**
- **TOUT: Timeout Error Interrupt Mask**
- **PECERR: PEC Error Interrupt Mask**
- **SMBDAM: SMBus Default Address Match Interrupt Mask**
- **SMBHHM: SMBus Host Header Address Match Interrupt Mask**

43.7.12 TWIHS Receive Holding Register

Name: TWIHS_RHR

Address: 0x40018030 (0), 0x4001C030 (1), 0x40060030 (2)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
RXDATA							

- **RXDATA: Master or Slave Receive Holding Data**

43.7.13 TWIHS SleepWalking Matching Register

Name: TWIHS_SWMR

Address: 0x4001804C (0), 0x4001C04C (1), 0x4006004C (2)

Access: Read/Write

31	30	29	28	27	26	25	24
DATAM							

23	22	21	20	19	18	17	16
–	SADR3						

15	14	13	12	11	10	9	8
–	SADR2						

7	6	5	4	3	2	1	0
–	SADR1						

- **SADR1: Slave Address 1**

Slave address 1. The TWIHS module matches on this additional address if SADR1EN bit is enabled.

- **SADR2: Slave Address 2**

Slave address 2. The TWIHS module matches on this additional address if SADR2EN bit is enabled.

- **SADR3: Slave Address 3**

Slave address 3. The TWIHS module matches on this additional address if SADR3EN bit is enabled.

- **DATAM: Data Match**

The TWIHS module extends the SleepWalking matching process to the first received data, comparing it with DATAM if DATAMEN bit is enabled.

43.7.14 TWIHS Transmit Holding Register

Name: TWIHS_THR

Address: 0x40018034 (0), 0x4001C034 (1), 0x40060034 (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
TXDATA							

- **TXDATA: Master or Slave Transmit Holding Data**

43.7.15 TWIHS Write Protection Mode Register

Name: TWIHS_WPMR

Address: 0x400180E4 (0), 0x4001C0E4 (1), 0x400600E4 (2)

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x545749 (“TWI” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x545749 (“TWI” in ASCII).

See Section 43.6.7 ”Register Write Protection” for the list of registers that can be write-protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x545749	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0

43.7.16 TWIHS Write Protection Status Register

Name: TWIHS_WPSR

Address: 0x400180E8 (0), 0x4001C0E8 (1), 0x400600E8 (2)

Access: Read-only

31	30	29	28	27	26	25	24
WPVSRC							

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last read of the TWIHS_WPSR.

1: A write protection violation has occurred since the last read of the TWIHS_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.