

# Interfacing EEPROM Using C2000 I2C Module



Manoj Kumar Santha Mohan

## ABSTRACT

This application note describes how C2000™ I2C can be used to communicate with EEPROM using both polling method (or) Interrupt method. This document implements different EEPROM write protocols such as Byte Write, Word Write, Paged Write operations and different EEPROM read operation such as Byte Read, Word Read, Paged Read operations.

EEPROM used for this example example is AT24C256 (which has write cycle time of 6ms and paged operation of 64 bytes)

**I2C EEPROM example code is available in below path:**

Polling method example: <C2000Ware>\driverlib\<device>\examples\c28x\i2c\i2c\_ex4\_eeprom\_polling

Interrupt method example: <C2000Ware>\driverlib\<device>\examples\c28x\i2c\i2c\_ex6\_eeprom\_interrupt

---

## Table of Contents

<b>1 Introduction</b> .....	<b>2</b>
<b>2 Hardware Connection</b> .....	<b>2</b>
<b>3 C2000 I2C Source Code</b> .....	<b>3</b>
3.1 I2CHandle Description.....	3
3.2 I2CBusScan.....	3
3.3 I2C_MasterTransmitter.....	4
3.4 I2C_MasterReceiver.....	4
<b>4 EEPROM Byte Write</b> .....	<b>5</b>
<b>5 EEPROM Byte Read</b> .....	<b>6</b>
<b>6 EEPROM Word Write</b> .....	<b>7</b>
<b>7 EEPROM Word Read</b> .....	<b>8</b>
<b>8 EEPROM Paged Write</b> .....	<b>9</b>
<b>9 EEPROM Paged Read</b> .....	<b>10</b>

## List of Figures

Figure 4-1. EEPROM Byte Write command.....	5
Figure 5-1. EEPROM Byte Read command.....	6
Figure 6-1. EEPROM Word Write Command.....	7
Figure 7-1. EEPROM Word Read Command.....	8
Figure 8-1. EEPROM Paged Write Command.....	9
Figure 9-1. EEPROM Paged Read Command.....	10

## List of Tables

Table 3-1. Project Filename and Descriptions.....	3
Table 3-2. I2C Interrupts Used in i2c_ex6_eeprom_interrupt Example.....	3
Table 3-3. Details of I2CHandle.....	3

## Trademarks

C2000™ is a trademark of Texas Instruments.  
All trademarks are the property of their respective owners.

## 1 Introduction

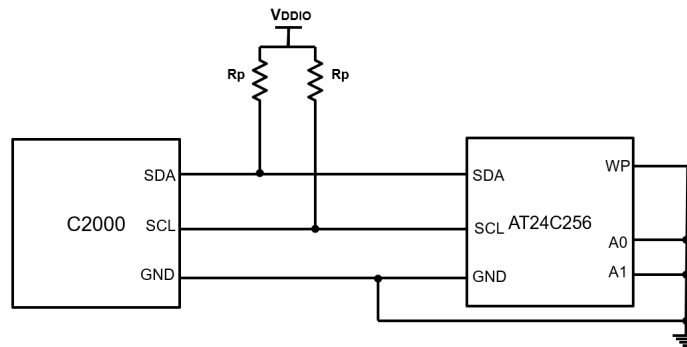
The C28x-I2C module used in the application note has the following features:

- Compliance with the NXP Semiconductors I2C bus specification (version 2.1):
  - Support for 8-bit format transfers
  - 7-bit and 10-bit addressing modes
  - General call
  - START byte mode
  - Support for multiple master-transmitters and slave-receivers
  - Support for multiple slave-transmitters and master-receivers
  - Combined master transmit/receive and receive/transmit mode
  - Data transfer rate from 10 kbps up to 400 kbps (Fast-mode)
- Receive FIFO and Transmitter FIFO (16-deep x 8-bit FIFO)
- Supports two ePIE interrupts:
  - I2Cx Interrupt – Any of the below events can be configured to generate an I2Cx interrupt:
    - Transmit-data ready
    - Receive-data ready
    - Register-access ready
    - No-acknowledgment received
    - Arbitration lost
    - Stop condition detected
    - Addressed as slave
  - I2Cx\_FIFO interrupts:
    - Transmit FIFO interrupt
    - Receive FIFO interrupt
- Module enable/disable capability
- Free data format mode

## 2 Hardware Connection

The below schematics shows how an EEPROM device can be connected to C2000 I2C module. EEPROM used in this application report is AT24C256. In AT24C256, user configurable pins (A0, A1) called device address pins can be used to address as many as four AT24C256 devices on the same I2C bus. These A0, A1 pins are pulled down which makes the slave address of EEPROM = 0x50. The write protect input pin needs to be connected to ground to allow EEPROM write operation.

For information regarding selection of pull resistor, see the [I2C Bus Pull-Up Resistor Calculation](#).



### 3 C2000 I2C Source Code

The C2000Ware software example provided in [Table 3-1](#) shows how to use I2C module to communicate with EEPROM via I2C bus. This example has been developed for EEPROM AT24C256, which requires 2 bytes for addressing the EEPROM memory with slave address of 0x50. [Table 3-2](#) shows the I2C interrupts used in EEPROM interrupt based example.

**Table 3-1. Project Filename and Descriptions**

Source Code	Description
i2c_ex4_eeprom_polling.c	This program will shows how to perform different EEPROM write and read commands using I2C polling method
i2cLib_FIFO_polling.c	C28x-I2C Library source file for FIFO using polling
i2cLib_FIFO_polling.h	C28x-I2C Library header file for FIFO using polling
i2c_ex6_eeprom_interrupt.c	This program will shows how to perform different EEPROM write and read commands using I2C interrupt method
i2cLib_FIFO_master_interrupt.c	C28x-I2C Library source file for FIFO interrupts
i2cLib_FIFO_master_interrupt.h	C28x-I2C Library header file for FIFO interrupts

**Table 3-2. I2C Interrupts Used in i2c\_ex6\_eeprom\_interrupt Example**

STOP condition	Register Access Ready
Addressed as slave	TX FIFO interrupt
Arbitration lost	RX FIFO interrupt
NACK condition	

#### 3.1 I2CHandle Description

[Table 3-3](#) provides the details of I2CHandle structure defined in the example code.

**Table 3-3. Details of I2CHandle**

Structure Members	Description
base	Base address of I2C module used
SlaveAddr	Slave address of EEPROM
pControlAddr	Pointer to variable which stores EEPROM address
NumOfAddrBytes	Number of address byte required by EEPROM
pTX_MsgBuffer	Pointer to TX message buffer
pRX_MsgBuffer	Pointer to RX message buffer
NumOfDataBytes	Number of data bytes in a I2C transaction
currentHandlePtr	Pointer to I2CHandle object
numofSixteenByte	Number of 16 bytes in data packet Note: FIFO depth in 16. so, we calculate number of 16 bytes.
remainingbytes	Number of bytes which are less than 16 bytes
WriteCycleTime_in_us	Write cycle time (in us) defined based on EEPROM

#### 3.2 I2CBusScan

**Function name:** I2CBusScan(I2CA\_BASE, pAvailableI2C\_slaves)

**Description:** This function can be used to scans for all I2C device connected to I2C bus by sending different slave addresses and checking for ACK / NACK condition

**Arguments:**

base : base is the base address of the I2C instance used


pAvailableI2C\_slaves : Address of the array that stores the I2C addresses available on the I2C bus

**Return:**Status of I2C transaction

```

16-Bit Hex - TI Style
0x0000A89C AvailableI2C_slaves
0x0000A89C 0050 0060 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x0000A8AC 0000 0000 0000 0000

```



### 3.3 I2C\_MasterTransmitter

**Function name:** I2C\_MasterTransmitter(I2CA\_BASE, struct I2CHandle \*pI2C\_Params)

**Description:** This function can be used to transmit 'N' number of bytes defined in pI2C\_Params. This function can be used to perform following EEPROM write operations.

1. EEPROM byte write  
Set EEPROM.NumOfDataBytes = 1, because one byte needs to be transmitted
2. EEPROM word write  
Set EEPROM.NumOfDataBytes = 2, because two bytes needs to be transmitted
3. EEPROM Paged write  
Set EEPROM.NumOfDataBytes = 'N', because 'N' bytes needs to be transmitted - Page size depends upon EEPROM chosen

Below setting applies to all the above EEPROM write operations

Set EEPROM.NumOfAddrBytes = 2, because high address byte and low address byte needs to be transmitted

**Arguments:**

base : base is the base address of the I2C instance used

pI2C\_Params : Pointer to I2C address handle of the slave

**Return:**Status of I2C transaction

### 3.4 I2C\_MasterReceiver

**Function name:** I2C\_MasterReceiver(I2CA\_BASE, struct I2CHandle \*pI2C\_Params)

**Description:** This function can be used to receive 'N' number of bytes defined in pI2C\_Params. This function can also be used to perform the following EEPROM read operations.

1. EEPROM byte read  
Set Number of bytes to be read (EEPROM.NumOfDataBytes = 1)
2. EEPROM word read  
Set Number of bytes to be read (EEPROM.NumOfDataBytes = 2) - Two bytes make a word
3. EEPROM Paged read  
Set Number of bytes to be read (EEPROM.NumOfDataBytes = 'N') . There is no page size restriction for paged read.

Below setting applies to all the above EEPROM read operations

Set EEPROM.NumOfAddrBytes = 2, because high address byte and low address byte needs to be transmitted

**Arguments:**

base : base is the base address of the I2C instance used

pI2C\_Params : Pointer to I2C address handle of the slave

**Return:**Status of I2C transaction

## 4 EEPROM Byte Write

Figure 4-1 shows how EEPROM Byte Write protocol is defined in AT24C256. C2000 I2C is configured in Master Transmitter mode (I2CMDR.MST = 1, I2CMDR.TRX = 1) to transmit EEPROM address (both High address byte, Low address byte) followed by data byte.



Figure 4-1. EEPROM Byte Write command

Code flow:

1. START condition + Transmit Slave address (0x50) + Write bit + ACK bit (from slave)
2. Transmit EEPROM high address byte + ACK bit (from slave)
3. Transmit EEPROM low address byte + ACK bit (from slave)
4. Transmit data byte + ACK bit (from slave)
5. Generate STOP condition

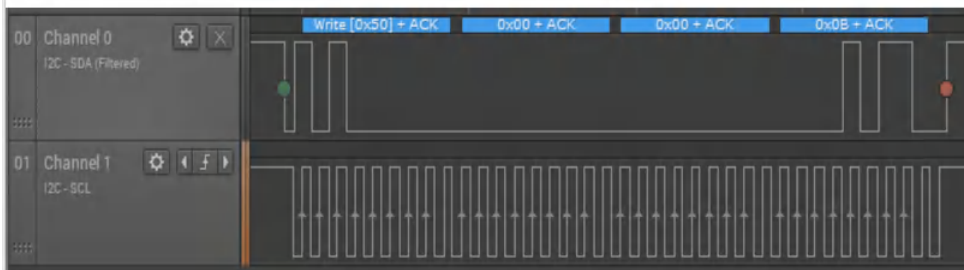
```
EEPROM.SlaveAddr = 0x50;
EEPROM.base = I2CA_BASE;
EEPROM.pControlAddr = &ControlAddr;
EEPROM.NumOfAddrBytes = 2;
EEPROM.pTX_MsgBuffer = TX_MsgBuffer;
EEPROM.pRX_MsgBuffer = RX_MsgBuffer;
EEPROM.NumOfAttempts = 5;
EEPROM.Delay_us = 10;
EEPROM.WriteCycleTime_in_us = 6000;
```

```
//Example 1: EEPROM Byte Write
//Write 11 to EEPROM address 0x0
```

```
ControlAddr = 0;
EEPROM.NumOfDataBytes = 1;
TX_MsgBuffer[0] = 11;
status = I2C_MasterTransmitter(&EEPROM);
```

Depends upon EEPROM used. Refer EEPROM datasheet

```
//Wait for EEPROM write cycle time
//This delay is not mandatory. User can run their application code
//It is however important to wait for EEPROM write cycle time before
//another read / write transaction
DEVICE_DELAY_US(EEPROM.WriteCycleTime_in_us);
```



9

## 5 EEPROM Byte Read

Figure 5-1 shows how EEPROM Byte Read protocol is defined in AT24C256. C2000 I2C is configured in Master Transmitter mode (I2CMR.MST = 1, I2CMR.TRX = 1) to transmit EEPROM address (both High address byte, Low address byte) and then C2000 I2C generates Repeated START condition in Master Receiver (I2CMR.MST = 1, I2CMR.TRX = 0) mode to receive a data byte from EEPROM.

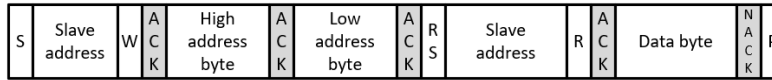


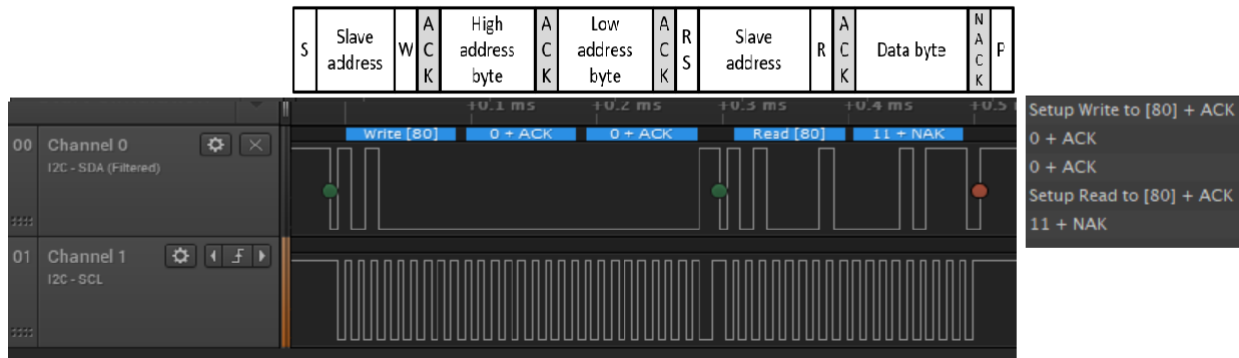
Figure 5-1. EEPROM Byte Read command

Code flow:

1. START condition + Transmit Slave address (0x50) + Write bit + ACK bit (from slave)
2. Transmit EEPROM high address byte + ACK bit (from slave)
3. Transmit EEPROM low address byte + ACK bit (from slave)
4. Repeated START condition + Transmit Slave address (0x50) + Read bit + ACK bit (from slave)
5. Receive data byte + ACK bit (from master)
6. Generate STOP condition

```
//Example 2: EEPROM Byte Read
//Make sure 11 is written to EEPROM address 0x0
ControlAddr = 0;
EEPROM.pControlAddr = &ControlAddr;
EEPROM.NumOfDataBytes = 1;
status = I2C_MasterReceiver(&EEPROM);

while(I2C_getStatus(EEPROM.base) & I2C_STS_BUS_BUSY);
```



## 6 EEPROM Word Write

Figure 6-1 shows how EEPROM Word Write protocol is defined in AT24C256. C2000 I2C is configured in Master Transmitter mode (I2CMDR.MST = 1, I2CMDR.TRX = 1) to transmit EEPROM address (both High address byte, Low address byte) followed by two data bytes.

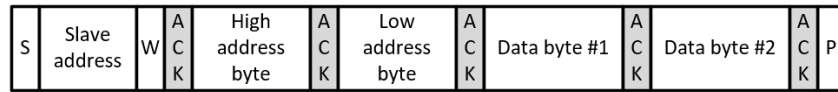


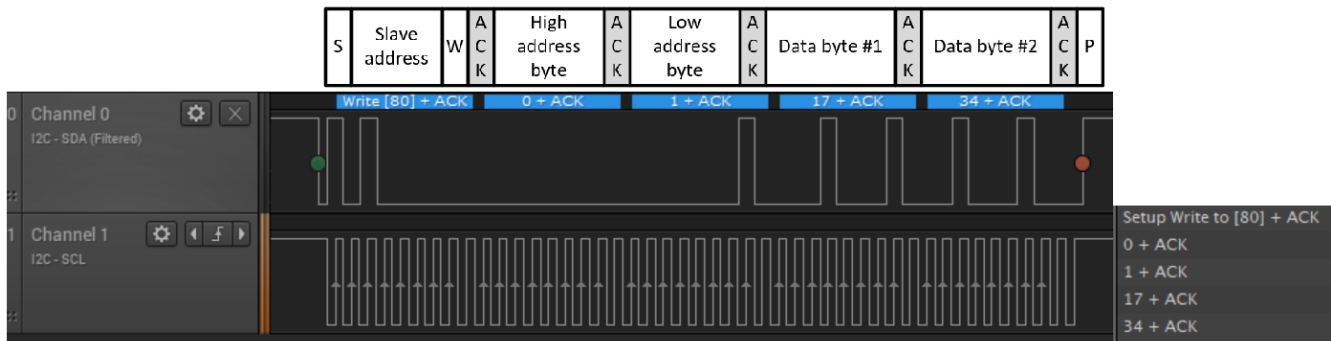
Figure 6-1. EEPROM Word Write Command

Code flow:

1. START condition + Transmit Slave address (0x50) + Write bit + ACK bit (from slave)
2. Transmit EEPROM high address byte + ACK bit (from slave)
3. Transmit EEPROM low address byte + ACK bit (from slave)
4. Transmit data byte # 1 + ACK bit (from slave)
5. Transmit data byte # 2 + ACK bit (from slave)
6. Generate STOP condition

```
//Example 3: EEPROM word (16-bit) write
//EEPROM address 0x1 = 22 & 0x2 = 33
ControlAddr = 1; //EEPROM address to write
EEPROM.NumOfDataBytes = 2;
TX_MsgBuffer[0] = 0x11;
TX_MsgBuffer[1] = 0x22;
EEPROM.pTX_MsgBuffer = TX_MsgBuffer;
status = I2C_MasterTransmitter(&EEPROM);

//Wait for EEPROM write cycle time
//This delay is not mandatory. User can run their application code instead.
//It is however important to wait for EEPROM write cycle time before you initiate
//another read / write transaction
DEVICE_DELAY_US(EEPROM.WriteCycleTime_in_us);
```



## 7 EEPROM Word Read

Figure 7-1 shows how EEPROM Word Read protocol is defined in AT24C256. C2000 I2C is configured in Master Transmitter mode (I2CMR.MST = 1, I2CMR.TRX = 1) to transmit EEPROM address (both High address byte, Low address byte) and then C2000 I2C generates Repeated START condition in Master Receiver (I2CMR.MST = 1, I2CMR.TRX = 0) mode to receive two data bytes from EEPROM.

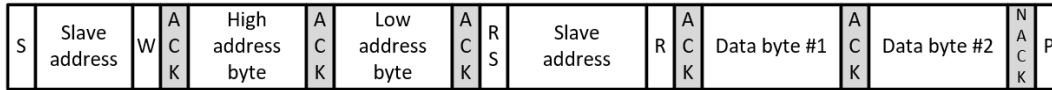


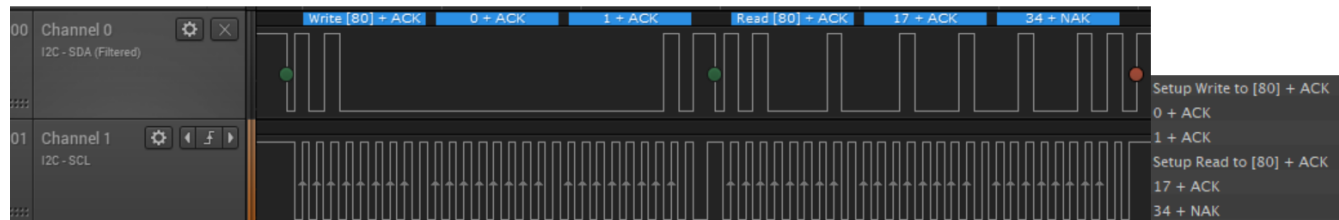
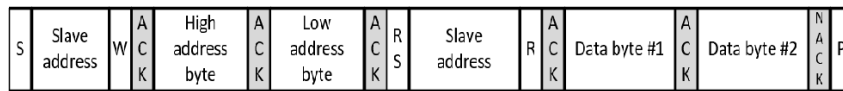
Figure 7-1. EEPROM Word Read Command

Code flow:

1. START condition + Transmit Slave address (0x50) + Write bit + ACK bit (from slave)
2. Transmit EEPROM high address byte + ACK bit (from slave)
3. Transmit EEPROM low address byte + ACK bit (from slave)
4. Generate Repeated START condition + Transmit Slave address (0x50) + Read bit + ACK bit (from slave)
5. Receive data byte # 1 + ACK bit (from master)
6. Receive data byte # 2 + NACK bit (from master)
7. Generate STOP condition

```
//Example 4: EEPROM word (16-bit) read
//Make sure EEPROM address 1 has 0x11 and 2 has 0x22
ControlAddr = 1;
EEPROM.pControlAddr = &ControlAddr;
EEPROM.pRX_MsgBuffer = RX_MsgBuffer;
EEPROM.NumOfDataBytes = 2;
```

```
status = I2C_MasterReceiver(&EEPROM);
```





## 8 EEPROM Paged Write

Figure 8-1 shows how EEPROM Paged Write protocol is defined in AT24C256. C2000 I2C is configured in Master Transmitter mode (I2CMDR.MST = 1, I2CMDR.TRX = 1) to transmit EEPROM address (both High address byte, Low address byte) followed by 'N' data bytes.

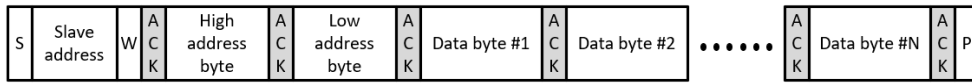


Figure 8-1. EEPROM Paged Write Command

Code flow:

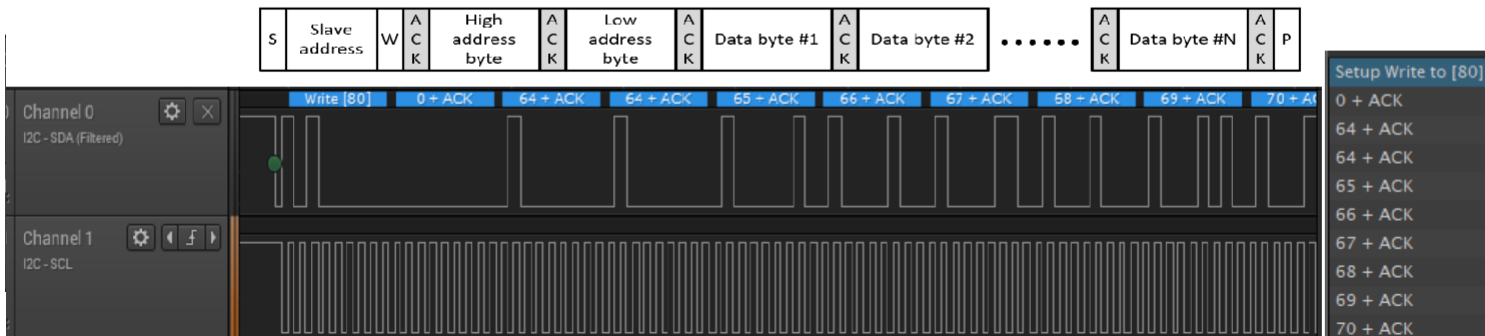
1. START condition + Transmit Slave address (0x50) + Write bit + ACK bit (from slave)
2. Transmit EEPROM high address byte + ACK bit (from slave)
3. Transmit EEPROM low address byte + ACK bit (from slave)
4. Transmit data byte # 1 + ACK bit (from slave)
5. Transmit data byte # 2 + ACK bit (from slave)
- ooo (Transmit more bytes)
6. Transmit data byte # N + ACK bit (from slave)
7. Generate STOP condition

```
//Example 5: EEPROM Page write
//Program address = data pattern from address 64

for(i=0;i<MAX_BUFFER_SIZE;i++)
{
    TX_MsgBuffer[i] = i+64;
}

ControlAddr = 64; //EEPROM address to write
EEPROM.NumOfDataBytes = MAX_BUFFER_SIZE;
EEPROM.pTX_MsgBuffer = TX_MsgBuffer;
status = I2C_MasterTransmitter(&EEPROM);

//Wait for EEPROM write cycle time
//This delay is not mandatory. User can run their application code instead.
//It is however important to wait for EEPROM write cycle time before you initiate
//another read / write transaction
DEVICE_DELAY_US(EEPROM.WriteCycleTime_in_us);
```



## 9 EEPROM Paged Read

Figure 9-1 shows how EEPROM Paged Read protocol is defined in AT24C256. C2000 I2C is configured in Master Transmitter mode (I2CMR.MST = 1, I2CMR.TRX = 1) to transmit EEPROM address (both High address byte, Low address byte) and then C2000 I2C generates Repeated START condition in Master Receiver (I2CMR.MST = 1, I2CMR.TRX = 0) mode to receive 'N' data bytes from EEPROM.



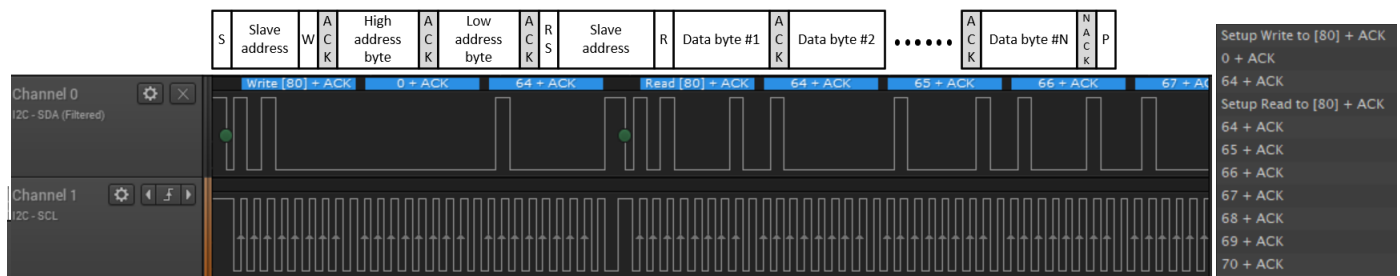
Figure 9-1. EEPROM Paged Read Command

Code flow:

1. START condition + Transmit Slave address (0x50) + Write bit + ACK bit (from slave)
2. Transmit EEPROM high address byte + ACK bit (from slave)
3. Transmit EEPROM low address byte + ACK bit (from slave)
4. Generate Repeated START condition + Transmit Slave address (0x50) + Read bit + ACK bit (from slave)
5. Receive data byte # 1 + ACK bit (from master)
6. Receive data byte # 2 + NACK bit (from master)
- o o o (Receive more bytes)
7. Receive data byte # N + ACK bit (from master)
8. Generate STOP condition

```
//Example 6: EEPROM word Paged read
ControlAddr = 64;
EEPROM.pControlAddr = &ControlAddr;
EEPROM.pRX_MsgBuffer = RX_MsgBuffer;
EEPROM.NumOfDataBytes = MAX_BUFFER_SIZE;

status = I2C_MasterReceiver(&EEPROM);
```



## IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to [TI's Terms of Sale](#) or other applicable terms available either on [ti.com](http://ti.com) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2022, Texas Instruments Incorporated