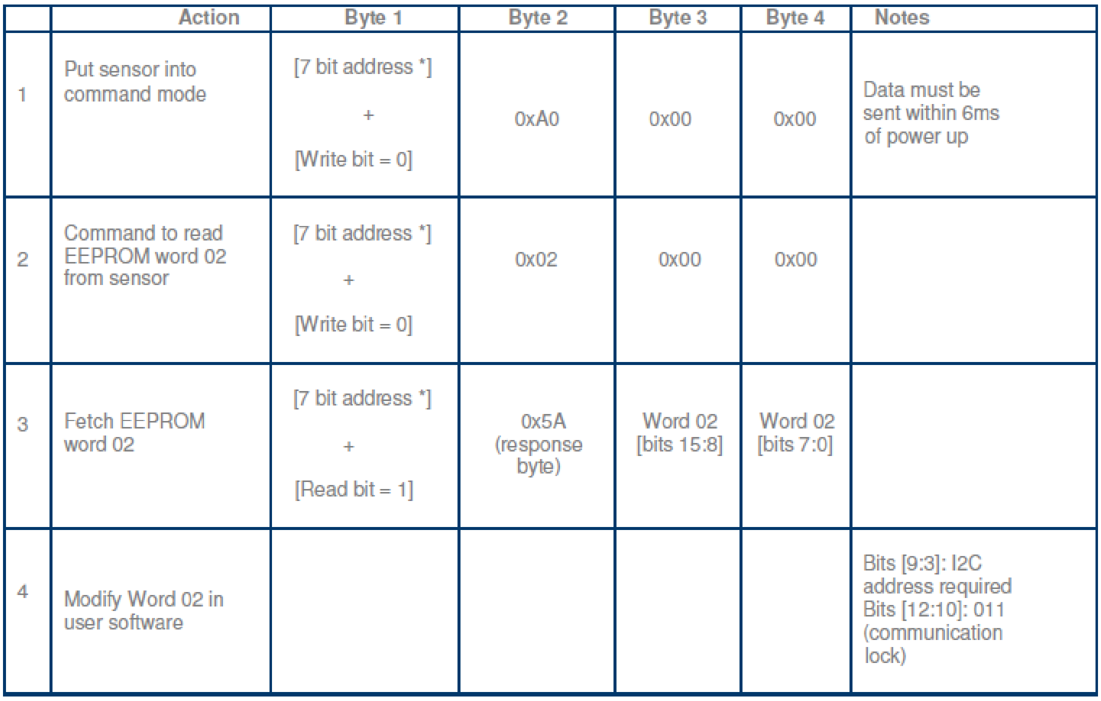
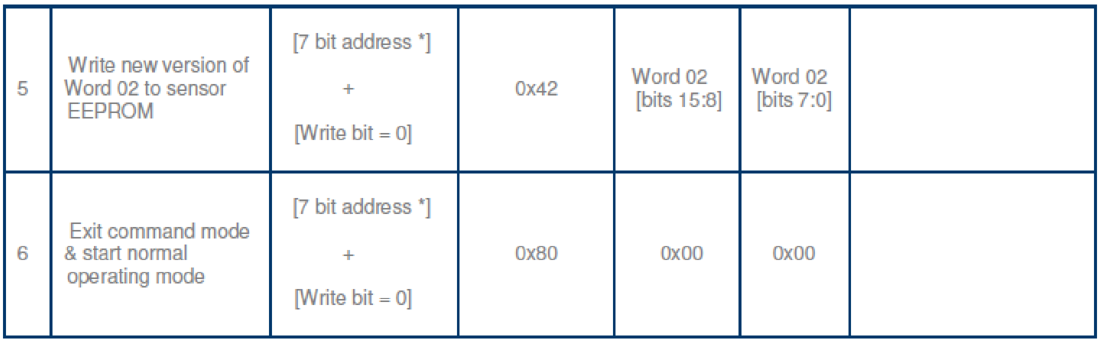
MCT-Series Application Note

CHANGING I2C SLAVE ADDRESS

In certain technical applications, the necessity to change the I2C address of pressure sensors arises from practical considerations and system design requirements. Pressure sensors, crucial components in various industrial and electronic systems, often communicate with microcontrollers or other devices using the I2C protocol. In scenarios where multiple sensors coexist within the same network, having unique I2C addresses becomes imperative to prevent address conflicts and ensure seamless communication. Address customization facilitates the integration of multiple sensors into a single system, allowing for efficient data acquisition and processing. Moreover, changing the I2C address provides the flexibility to configure sensor networks as per the specific needs of a project, optimizing resource utilization and enabling more sophisticated and scalable sensor deployments. Therefore, the ability to modify I2C addresses on pressure sensors is a valuable feature that enhances the adaptability and versatility of sensor networks in diverse applications; this application note address this need.

Table 1: Changing I2C Slave Address





	Action	Byte 1	Byte 2	Byte 3	Byte 4	Notes
1	Put sensor into command mode	[7 bit address *] + [Write bit = 0]	0xA0	0x00	0x00	Data must be sent within 6ms of power up
2	Command to read EEPROM word 02 from sensor	[7 bit address *] + [Write bit = 0]	0x02	0x00	0x00	
3	Fetch EEPROM word 02	[7 bit address *] + [Read bit = 1]	0x5A (response byte)	Word 02 [bits 15:8]	Word 02 [bits 7:0]	
4	Modify Word 02 in user software					Bits [9:3]: I2C address required Bits [12:10]: 011 (communication lock)
5	Write new version of Word 02 to sensor EEPROM	[7 bit address *] + [Write bit = 0]	0x42	Word 02 [bits 15:8]	Word 02 [bits 7:0]	
6	Exit command mode & start normal operating mode	[7 bit address *] + [Write bit = 0]	0x80	0x00	0x00	

C Code Example

The following code changes the I2C slave address of MS45x5DO sensor.

```
#define OldI2Caddress 0x28
#define NewI2Caddress 0x36
unsigned char EEPROMdata[4];
unsigned int tempEEPROMdata;
bool SetNewI2Caddress(unsigned char OldAddress,unsigned char NewAddress);
bool EnterCommandMode(unsigned char I2Caddress);
bool ReadEEPROM (unsigned I2Caddress,unsigned char EEPROMaddress);
bool FetchEEPROM (unsigned char Address,unsigned char Quantity,unsigned char *Data);
bool WriteDataToSensor (unsigned char Address, unsigned char Quantity, unsigned char *Data);
void START(void);
void STOP(void);
bool DetectACK(void);
void SendACK(void);
void main (void)
{
SetNewI2Caddress(OldI2Caddress, NewI2Caddress);
for(;;)
}
bool SetNewI2Caddress(unsigned char OldAddress,unsigned char NewAddress)
{
bool result=false;
PowerUpSensor (); //Power up MS45x5DO sensor
Delay (3); //Delay 3ms.
if (EnterCommandMode(OldI2Caddress)==true) //Put sensor into command mode.
{
if (ReadEEPROM(OldI2Caddress,0x02)==true) //Read EEPROM word 02
{
if (FetchEEPROM(OldI2Caddress,3, EEPROMdata)==true) //Fetch EEPROM word 02
{
if (EEPROMdata [0]==0x5A)
{
tempEEPROMdata= (((EEPROMdata [1] << 8) + EEPROMdata [2]) & 0xE007) + (NewI2Caddress << 3) + 0xC00;
EEPROMdata [1] = (unsigned char) ((tmpEEPROMdata & 0xff00)>>8);
EEPROMdata [2] = (unsigned char) (tmpEEPROMdata & 0x00ff);
EEPROMdata [0] =0x42;
if (WriteDataToSensor(OldI2Caddress,3, EEPROMdata)==true) //Write new version of //Word 02 to sensor //EEPROM
{
EEPROMdata [0]=0x80;
EEPROMdata [1]=0x00;
EEPROMdata [2]=0x00;
if (WriteDataToSensor (OldI2Caddress,3, EEPROMdata)==true) //Exit command mode & //start
normal operating
//mode.
{
result=true;
}
}
}
}
}
}
return result;
}
bool EnterCommandMode(unsigned char I2Caddress)
{
EEPROMdata [0] = 0xA0;
EEPROMdata [1] = 0x00;
EEPROMdata [2] = 0x00;
return (WriteDataToSensor (I2Caddress,3, EEPROMdata));
}
bool ReadEEPROM (unsigned I2Caddress,unsigned char EEPROMaddress)
{
EEPROMdata [0] = EEPROMaddress;
EEPROMdata [1] = 0x00;
EEPROMdata [2] = 0x00;
return (WriteDataToSensor (I2Caddress,3, EEPROMdata));
}
bool FetchEEPROM (unsigned char Address,unsigned char Quantity,unsigned char *Data)
{
unsigned char index;
START();
Address=(Address<<1)+0x01;
SendOneByteData(Address); //send address..
if (DetectACK()==false) //check ACK.
{
STOP();
return false;
```

```
}
//
For (index=0; index <Quantity;Index++)
{
Data[index]=ReadOneByteData();
if (index<(Quantity-1))
{
SendACK();
        }
}
STOP();
return true;
}

bool WriteDataToSensor (unsigned char Address, unsigned char Quantity, unsigned char *Data)
{
unsigned char index;
START();
.
SendOneByteData(Address<<1); //send address and Write Command
if (DetectACK()==false) //check ACK.
{
STOP();
return false;
}
for(index=0;index<Quantity;index++)
{
SendOneByteData (Data[index]);
if (DetectACK()==false)
{
STOP();
return false;
}
}
STOP();
return true;
}
```