ECM AFM9E: Introductory* Application Guide

The AFM9E is a universal exhaust gas oxygen (UEGO) sensor controller. The user can interface with the AFM9E through a serial peripheral interface (SPI) or an analog input. Through the SPI, the user can calibrate the sensor and read values for oxygen and λ. If using the analog interface, the user calculates the value of λ or oxygen by using the lookup table provided.

Section 2 shows how to connect the sensor for both configurations. Sections 3 and 4 describe how to get started with the analog interface. Sections 5 through 10 show how to get started using the SPI interface. Section 11 contains reference information.
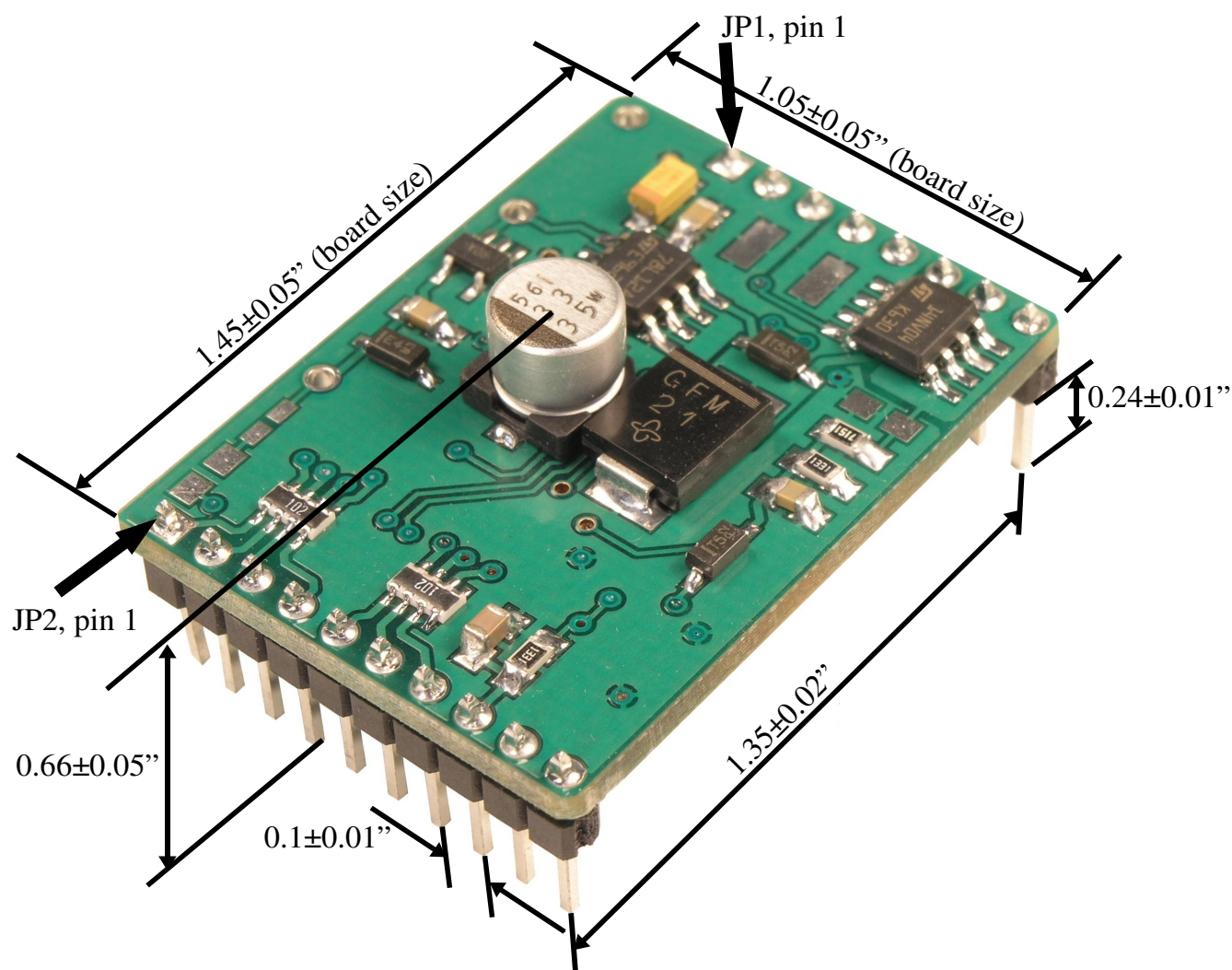
1. Hardware Dimensions

Figure 1. Hardware Dimensions

* This document is a simplified, introductory user's guide to the AFM9E. More detailed application-specific guides are available.

2. Connecting the Sensor

The sensor is connected to the AFM9E as shown in Figure 2. Table 1 lists the wiring specifications for connecting the sensor to the AFM9E.
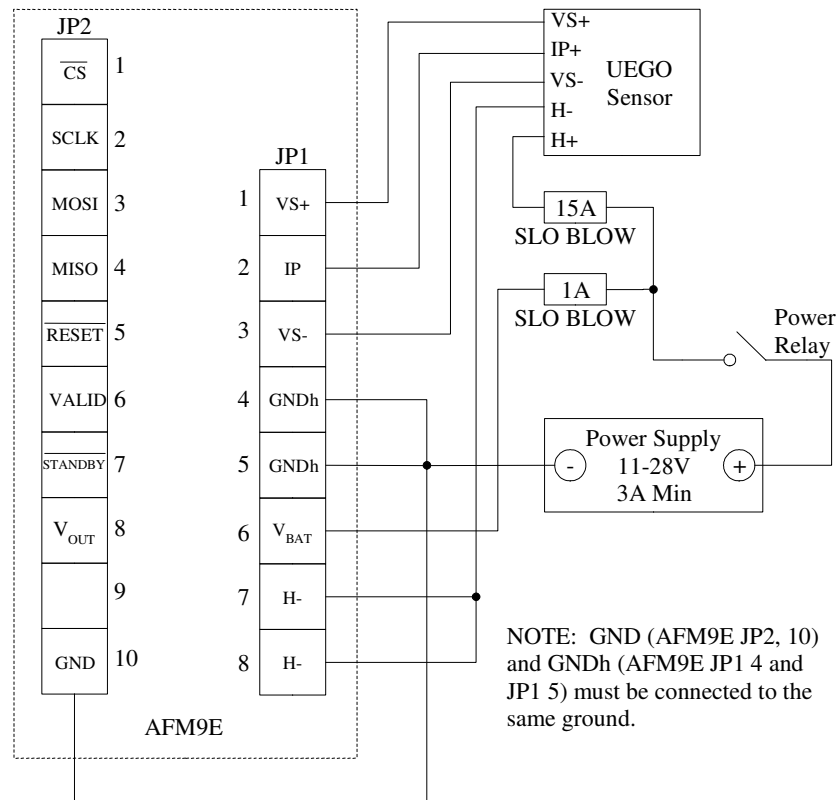

Figure 2. Sensor Connections

To	From	Bosch Wire Color	NTK Wire Color	Min Wire Size	Max Wire Length*
AFM9E JP1,1	Sensor VS+	Black	Grey	22 AWG	20 feet
AFM9E JP1,2	Sensor IP+	Red	White	22 AWG	20 feet
AFM9E JP1,3	Sensor VS-	Yellow	Black	22 AWG	20 feet
AFM9E JP1,7	Sensor H-	White	Yellow	20 AWG	20 feet
AFM9E JP1,8	Sensor H-	White	Yellow	20 AWG	20 feet
AFM9E JP1,4	Power Supply (NEG)	NA	NA	20 AWG	20 feet
AFM9E JP1,5	Power Supply (NEG)	NA	NA	20 AWG	20 feet
AFM9E JP1,6	Power Supply (POS)	NA	NA	20 AWG	20 feet
AFM9E JP2,10	Power Supply (NEG)	NA	NA	20 AWG	20 feet
Sensor H+	Power Supply (POS)	Grey	Blue	20 AWG	20 feet

Table 1. Sensor Wiring Connections

*Longer lengths are possible with a lower gauge wire. Contact ECM.

3. Analog Microcontroller Configuration

In the analog configuration, the user microcontroller is connected with three wires as shown in Figure 3. Table 2 shows how to configure each pin on the user microcontroller. The voltage levels for VALID (V_{OH}, V_{OL}) and $\overline{STANDBY}$ (V_{IH}, V_{IL}) are defined in Section 11.
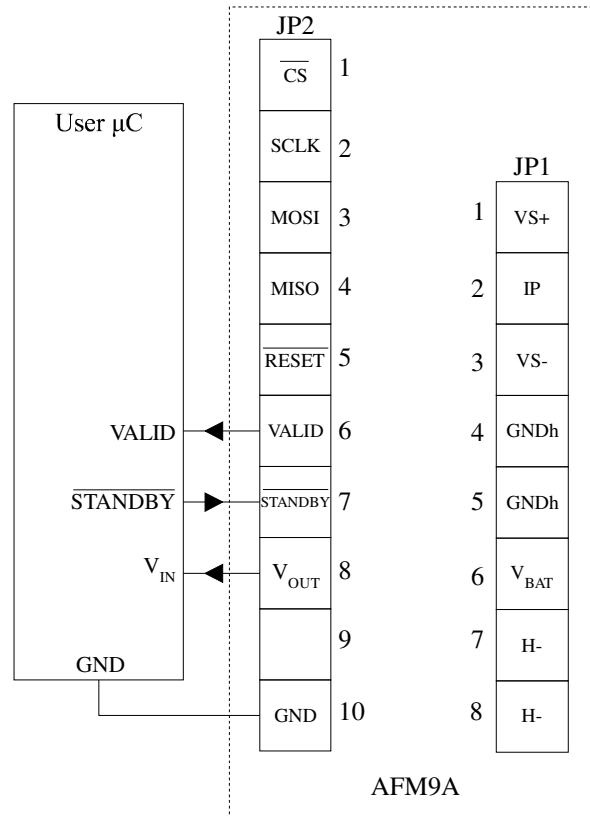

Figure 3. Analog Connections

AFM9E Pin	Symbol (AFM9E)	Symbol (User Microcontroller)	User Microcontroller Port Configuration
JP2, 6	VALID	VALID	General Purpose Digital Input (0 or 5V)
JP2, 7	$\overline{STANDBY}$	$\overline{STANDBY}$	General Purpose Digital Output (0 or 5V)
JP2, 8	V_{OUT}	V_{IN}	Analog Input (0 to 5V)
JP2, 10	GND	GND	Common Ground

Table 2. Analog Connections and Port Configuration

4. Analog Operation

When the AFM9E is powered on, it initializes and then warms-up the sensor. During initialization and warm-up, VALID is low and V_{OUT} corresponds to 0% oxygen. Once the sensor is ready, VALID is set high and V_{OUT} changes linearly with the measured value of oxygen. V_{OUT} can be used to read the value of λ or oxygen according to Table 9 in Section 11. If the user

microcontroller sets $\overline{\text{STANDBY}}$ to low, the AFM9E will set V_{OUT} to 0% oxygen, VALID to low, and turn the heater off (see Figure 9 in Section 11). VALID should be polled regularly; if it is low, there is an error. Error descriptions can be read via the SPI interface.

5. SPI Microcontroller Configuration

The AFM9E is connected to a SPI master as shown in Figure 4. The AFM9E operates in SPI slave mode 0 with an alternative implementation of the Slave Select called Chip Select (CS). Table 3 lists how each connection should be configured on the user microcontroller. All signals connected to the AFM9E should adhere to the V_{OH}, V_{OL}, V_{IH}, and V_{IL} values in Section 11.

SPI MASTER PERIPHERAL SETTINGS:

1. SPI MODE 0

2. CPOL = 0 (IDLE SCLK = LOW)

3. CPHA = 0 (Data valid on rising edge of SCLK)

4. LSBFE = 0 (MSB of data transmitted first)

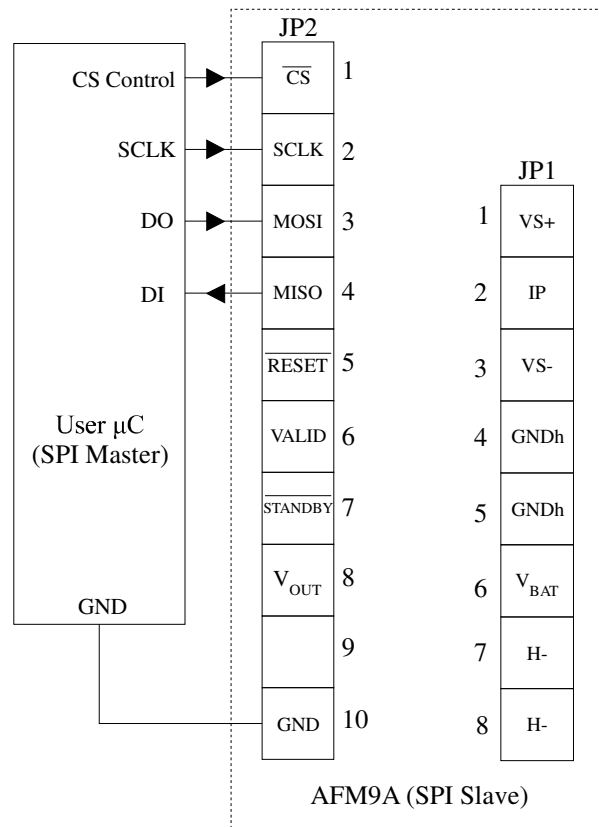5. MAX SCLK = 1.82MHz

6. USER uC is the SPI MASTER

Figure 4. SPI Connections

AFM9E Pin	Symbol (AFM9E)	Symbol (SPI Master)	SPI Master Port Configuration
JP2,1	$\overline{\text{CS}}$	CS Control	General Purpose Output
JP2,2	SCLK	SCLK	Serial Clock (Output)
JP2,3	MOSI	MOSI (DO)	Master Output/Slave Input (Output)
JP2,4	MISO	MISO (DI)	Master Input/Slave Output (Input)
JP2,10	GND	GND	Common Ground

Table 3. SPI Connections

6. SPI Protocol

The leading edge of the clock should be rising, the trailing edge should be falling, and when at idle the clock should be low. This requires CPOL to be set to zero. Set CPHA to zero so that data is read on the rising edge of the clock and changed on the falling edge. The MSB of the data is transmitted first; therefore set LBSFE to zero. The serial clock (SCLK) frequency can be a maximum of 1.82 MHz. The SPI should adhere to the timing diagrams in Figures 5 and 6.
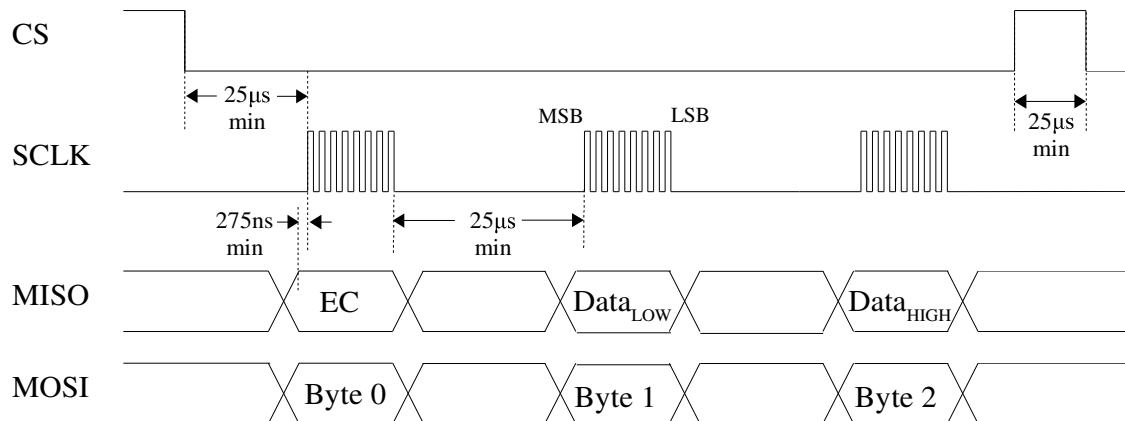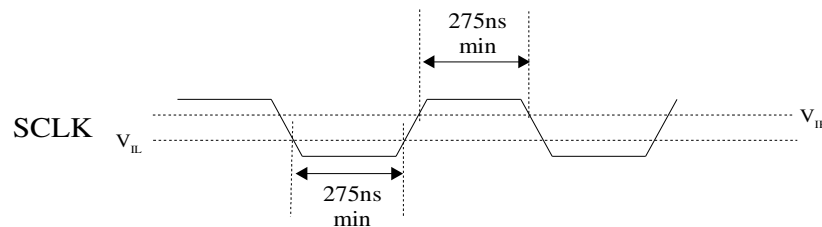

Figure 5. Timing Diagram

Figure 6. Serial Clock Timing Detail

The error code (EC) is automatically returned when the user microcontroller sends Byte 0 (see Figure 5). The possible values for the EC and their meanings are listed in Table 6.

The user microcontroller can read from or write to the AFM9E. Setting the most significant bit in Byte 0 to one indicates a write operation. The 7 least significant bits (b6-b0) indicate which 16-bit word on which to operate. A summary of common values for Byte 0 is in Table 4.

Byte 0 (b6-b0)	16-bit Word	Type
0x00	Command (read or write)	16-bit unsigned
0x01	Oxygen Value * 1000 (read only)	16-bit signed (two's complement)
0x02	λ Value * 1000 (read only)	16-bit unsigned

Table 4. SPI Protocol Summary

When writing to the AFM9E, the data bytes returned to the user microcontroller should be ignored. The AFM9E ignores the values of Byte 1 and Byte 2 during a read operation.

Command is used to perform special tasks. The high byte of Command is reserved. Writing a value to the low byte of Command starts a certain task. Reading from it returns the status of the task. Table 5 shows the meaning of each bit in the command word.

Bit	15	14	13	12	11	10	9	8
CommandH	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Read/Write	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
CommandL	Reserved	Reserved	ACFL	Reserved	WEE	Reserved	Reserved	PFAC
Read/Write	R	R	R	R	R/W	R	R	R/W
Initial Value	0	0	0	0	0	0	0	0

Table 5. Using Command

Bits 1, 2, 4, 6-15: Reserved. When writing to the command word, these bits should be set to zero.

Bit 5—ACFL: Air Calibration (Air Cal) Failed. If this bit is set to one, the last Air Cal attempt failed.

Bit 3—WEE: Write to EEPROM. Writing a one to this bit will save the current Air Cal to the EEPROM. Once the operation is complete, the AFM9E clears the flag.

Bit 0—PFAC: Perform Air Cal. Writing a one to this bit will initiate an Air Cal. Once the Air Cal is complete, the AFM9E clears this flag. If the Air Cal was unsuccessful, ACFL is set.

Error Code(EC)	Meaning	λ /Oxygen
0x00	Normal Operation	Valid
0x01	Initialization	Invalid
0x14	Sensor Heater Open	Invalid
0x15	Sensor Heater Shorted	Invalid
0x31	Low Voltage Warning ($V_{BAT} < 10.6V$)	Valid
0x32	High Voltage Warning ($V_{BAT} > 30V$)	Valid
0x41, 0x51	Sensor is Warming Up or Sensor Failure	Invalid
0x52	Not Enough Energy To Heat Sensor	Invalid

Table 6. Error Codes

7. Initializing and Running the AFM9E

After reset, the AFM9E initializes and then heats up the sensor. When the sensor is ready, the user can perform an optional air calibration (Air Cal) by placing the sensor in air and initializing a calibration. After the Air Cal is complete, it should be saved to EEPROM. The user can then

read values for oxygen or λ from the AFM9E. Figure 7 shows a software flow diagram to initialize and run the AFM9E.
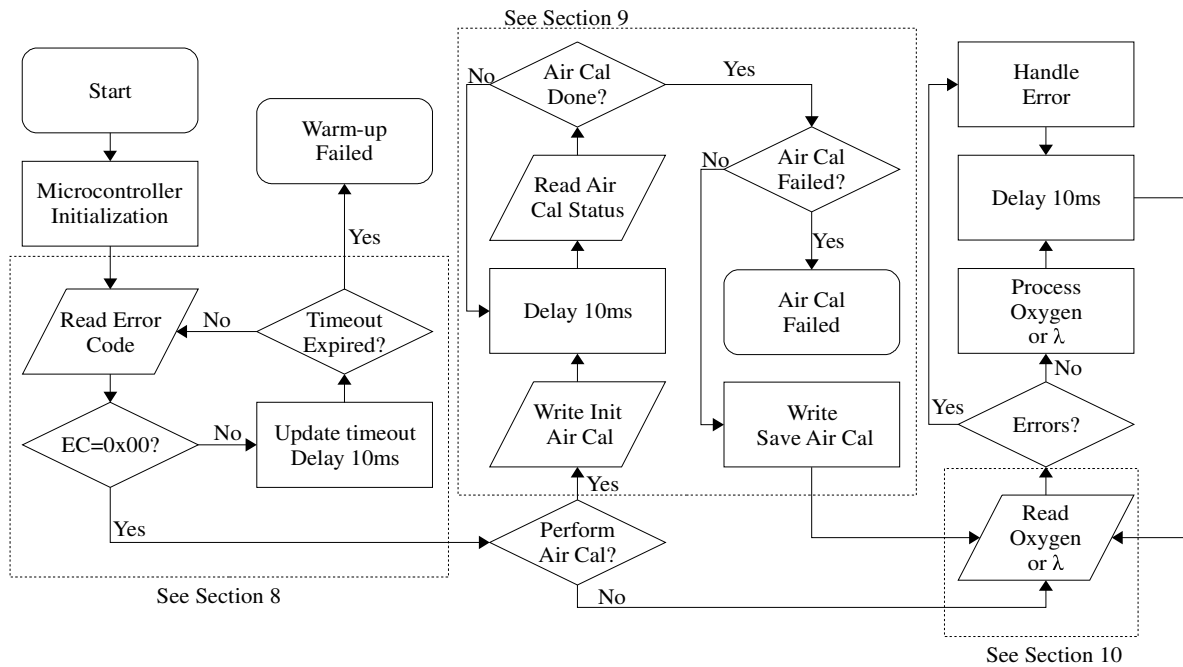

Figure 7. Software Flow Diagram

The microcontroller initialization should configure the user microcontroller to operate according to Sections 5 and 6. Section 8 has an example of how to monitor the sensor warm-up period. If the user wants to perform an Air Cal, Section 9 has an example of how to start the Air Cal by writing to Command, return the Air Cal status (success/failure) by reading Command, and save the Air Cal to EEPROM. If the Air Cal is not saved to EEPROM, the calibration will only be used until the next reset.

After the warm-up and optional Air Cal, the user can read values for oxygen or λ. To avoid resource conflicts, oxygen or λ should be read at most every 10ms. Section 10 has an example of how to read oxygen. If a non-zero EC is returned, refer to Table 6.

8. Software Example: AFM9E Initialization Sequence

The AFM9E boots from power-on reset into initialization. After initialization, the AFM9E warms-up the sensor. Table 7 shows the error codes that occur while initializing from power on to normal operation. When the AFM9E returns an error code of 0x00, the sensor is ready.

Error Code(EC)	Meaning
0x01	Initialization
0x41, 0x51	Sensor is Warming Up
0x00	Normal Operation

Table 7. Initialization and Warm-up Error Codes

```
/*************************************************
*   Sensor Warm-up Example
*************************************************/
char warmup(){
        char ec;                        //temporary variable for storing the error code
        int timer=0;                    //variable for a timeout
        while (timer < TIMEOUT){        //TIMEOUT should be about 6000 (60s/10ms)
                CLEAR_CS();             //user-defined function to clear the chip select signal
                delay_us(25);           //user-defined microsecond delay
                ec = SPItransfer(0x00); //user-defined spi operation
                SET_CS();               //user-defined function to set chip select signal high
                if (ec == 0x00) break;  //if the ec is 0x00, the sensor is ready
                timer++;
                delay_ms(10);           //wait 10ms between queries (user defined)
        }
        //if the sensor did not warm up properly in less than 60 seconds, there is a problem
        if (timer == TIMEOUT) return WARMUP_FAILED;
        else return WARMUP_SUCCESS;
}
```

9. Software Example: Performing an Air Calibration

To perform an Air Cal, the SPI master sends the sequence: 0x80 then 0x01 while the sensor is in ambient air. Bit 0 of Command will stay high until the calibration is complete. If there was an error, the calibration will abort and set bit 5 of Command. A successful calibration takes about 2 seconds.

```
/*************************************************
* Air Calibration Example
*************************************************/
char performAirCal(){
        char command=0x01;                      //sending 0x01 starts the Air Cal (see Table 5)
        CLEAR_CS();                             //function to clear Chip Select  (user-defined)
        delay_us(25);                           //Delay for 25us (user-defined)
        SPItransfer(0x80);                      //Prepare AFM9E to write command byte
        delay_us(25);                           //delay for 25us
        SPItransfer(command);                   //write 0x01 to command byte to start Air Cal
        SET_CS();                               //function to Set Chip Select high
        delay_us(25);
        while(command == 0x01){                 //wait until Air Cal is complete
                CLEAR_CS();                     //now read Command to determine Air Cal Status
                delay_us(25);
                SPItransfer(0x00);              //Prepare AFM9E to read Command
                delay_us(25);
                command = SPItransfer(0x00);    //store Command
                SET_CS();                       //Complete Transmission (only need the low byte)
                delay_ms(10);                   //user-defined ms delay
        }
        if (command==0x00){                     //if the Air Cal was successful…
                CLEAR_CS();
                delay_us(25);
                SPItransfer(0x80);              //Prepare AFM9E to write Command
```

```
            delay_us(25);
            SPItransfer(0x08);                  //Tell AFM9E to save Air Cal to EEPROM
            SET_CS();
        }
        return command; //0x00 for success and 0x20 (Bit 5 = 1) for a failed Air Cal (see Table 5)
                        //The Air Cal will fail if there are any non-zero error codes during the Air Cal
    }
```

10. Software Example: Reading Oxygen

To read oxygen, the SPI master sends 0x01 as Byte 0 (see Table 4), then two dummy bytes (0x00 in this example), and in return receives EC, O_{2LOW}, and O_{2HIGH} respectively (see Figure 8). Alternatively, the SPI master can send 0x02 as Byte 0 to read λ.

```
/************************************************
Read Oxygen Example
************************************************/
…
int oxygen;
char EC;
EC = readOxygen(&oxygen);          //to call the function pass the address of (&) the oxygen variable
…
char readOxygen(int * o2){
        unsigned char low, high;
        CLEAR_CS();                        //Function to clear Chip Select
        delay_us(25);                      //delay for 25 microseconds
        char ec = SPItransfer(0x01);       //prepare the AFM to send oxygen bytes
        delay_us(25);
        low = SPItransfer(0x00);           //low byte comes first (AFM9E ignores 0x00)
        delay_us(25);
        high = SPItransfer(0x00);          //high byte comes next (AFM9E ignores 0x00)
        SET_CS();                          //Transmission Complete (CS is set high)
        *o2 = high*256+low;                //calculate and store oxygen to specified address
        return ec;                         //return the error code
    }
```
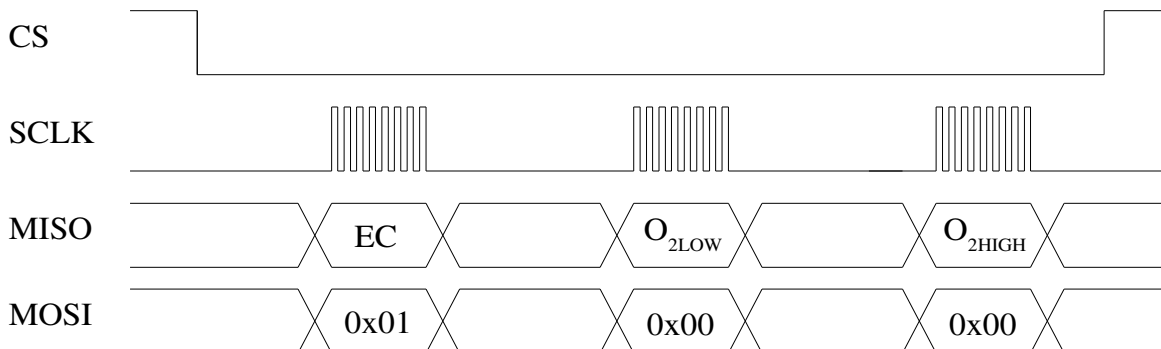

Figure 8. SPI Data Transfers for Reading Oxygen

The following equations convert the bytes received from the AFM9E to λ and percentage oxygen. If λ and oxygen values are invalid (see Table 6), the AFM9E will reply to requests for these values with 0 (0x0000) and -32768 (0x8000), respectively. Note that both λ and oxygen are scaled by a factor of 1000.

$$\lambda = \frac{\lambda_{HIGH} \cdot 256 + \lambda_{LOW}}{1000} \quad \text{16bit unsigned} \atop \text{(0 to 65.535 λ)} \qquad O_2 = \frac{O_{2HIGH} \cdot 256 + O_{2LOW}}{1000} \quad \text{16bit signed} \atop \text{(-32.768 to 32.767%)}$$

11. Reference

The input and output references in Table 8 are relative to the AFM9E.

Symbol	Parameter	Minimum	Typical	Maximum	Units
V_{BAT}	Battery (supply) Voltage	10.7	15	30	V
I_{BAT}	Battery (supply) Current	3			A
V_{IH}	Input High Voltage (Except Reset)	3		5.5	V
V_{IL}	Input Low Voltage	-0.5		1.5	V
V_{OH}	Output High Voltage	4.3			V
V_{OL}	Output Low Voltage			0.6	V
T	Operating Temperature	-40	25	85	°C

Table 8. Electrical Characteristics

| Index	VOUT	O2	λ	Index	VOUT	O2	λ	Index	VOUT	O2	λ
0	0.000	-28.220	0.462	9	1.646	-5.000	0.852	18	2.992	14.000	3.436
1	0.228	-25.000	0.516	10	1.823	-2.500	0.913	19	3.134	16.000	4.914
2	0.405	-22.500	0.559	11	2.000	0.000	1.000	20	3.205	17.000	6.205
3	0.582	-20.000	0.601	12	2.142	2.000	1.127	21	3.276	18.000	8.380
4	0.760	-17.500	0.640	13	2.284	4.000	1.285	22	3.311	18.500	10.133
5	0.937	-15.000	0.677	14	2.425	6.000	1.485	23	3.347	19.000	12.785
6	1.114	-12.500	0.715	15	2.567	8.000	1.747	24	3.382	19.500	17.266
7	1.291	-10.000	0.755	16	2.709	10.000	2.104	25	3.418	20.000	26.548
8	1.468	-7.500	0.800	17	2.851	12.000	2.622	26	3.485	20.950	AIR

Table 9. Nominal Oxygen and λ Lookup Table
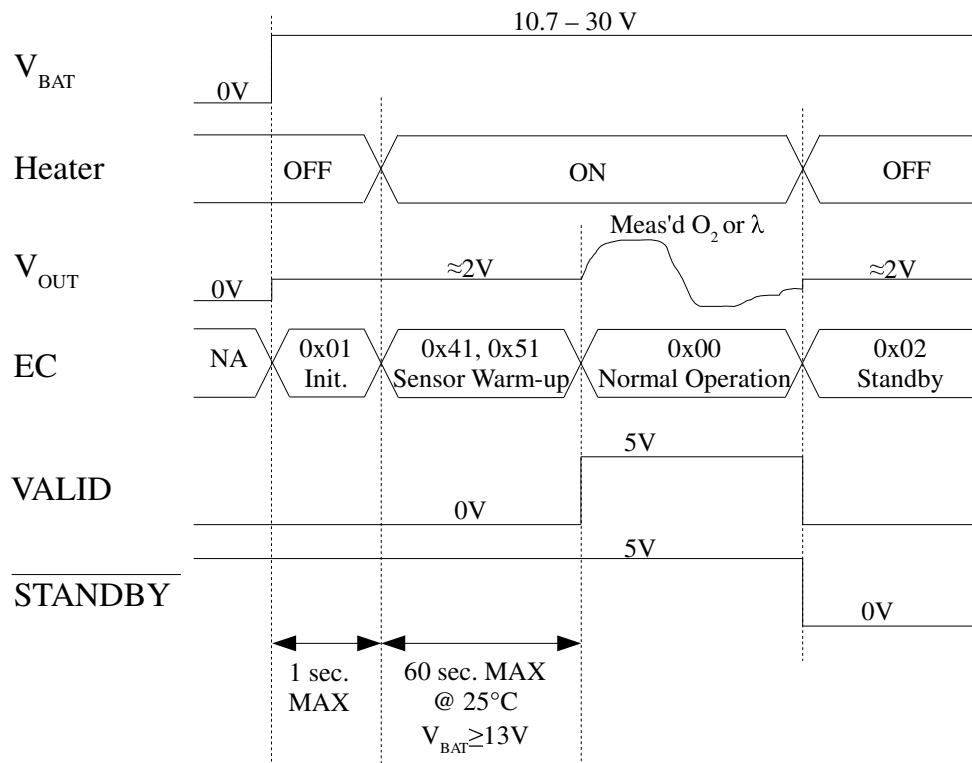
Figure 9. Sensor Start-up Sequence and Standby Mode

An air calibration can performed using the Vout (Pin 8 of JP2) in lieu of the SPI command. Connect the sensor to the AFM9E. Hang the sensor in free air and apply power to the AFM9E with STANDBY set high. Wait at least 5 minutes. Record the VOUT voltage as VCAL. Use the formula below to calculate an O2 index and use it to interpolate the correct lambda value in Table 9.

$$O2 = (VOUT – 2) / (VCAL – 2) * 20.95$$

Example: With a new sensor the AFM9E VOUT output reads 2.88V in free air. Calculate the lambda for an output voltage of 1.10V:

VCAL = 2.88V therefore O2 = (1.10 – 2) / (2.88-2) * 20.95 = -21.43
λ = (-21.43 – (-22.50)) / (-20.00 – (-22.50)) * (0.601 – 0.559) + 0.559 = 0.577

For assistance, contact ECM at 408-734-3433 between the hours of 10 am and 6 pm PST.

REV	DATE	DESCRIPTION	PAGE
3.7	3/23/2015	Original release	n/a