Freescale Semiconductor

KINETIS SDK I2C MASTER DRIVER

FRDMK64 – EEPROM TEST

By: Abigail Inzunza / Jorge Gonzalez

freescale semiconductor

KSDK I2C MASTER DRIVER
FRDMK64 – EEPROM TEST

About this document

The Kinetis Software Development Kit (KSDK) allows the users of Kinetis devices to simplify the development of different projects involving its different peripherals, such as the Inter-Integrated Circuit (I2C) module.

A basic I2C driver example to communicate a FRDMK-64F with an I2C EEPROM, developed using SDK, will be explained on this document.

The driver

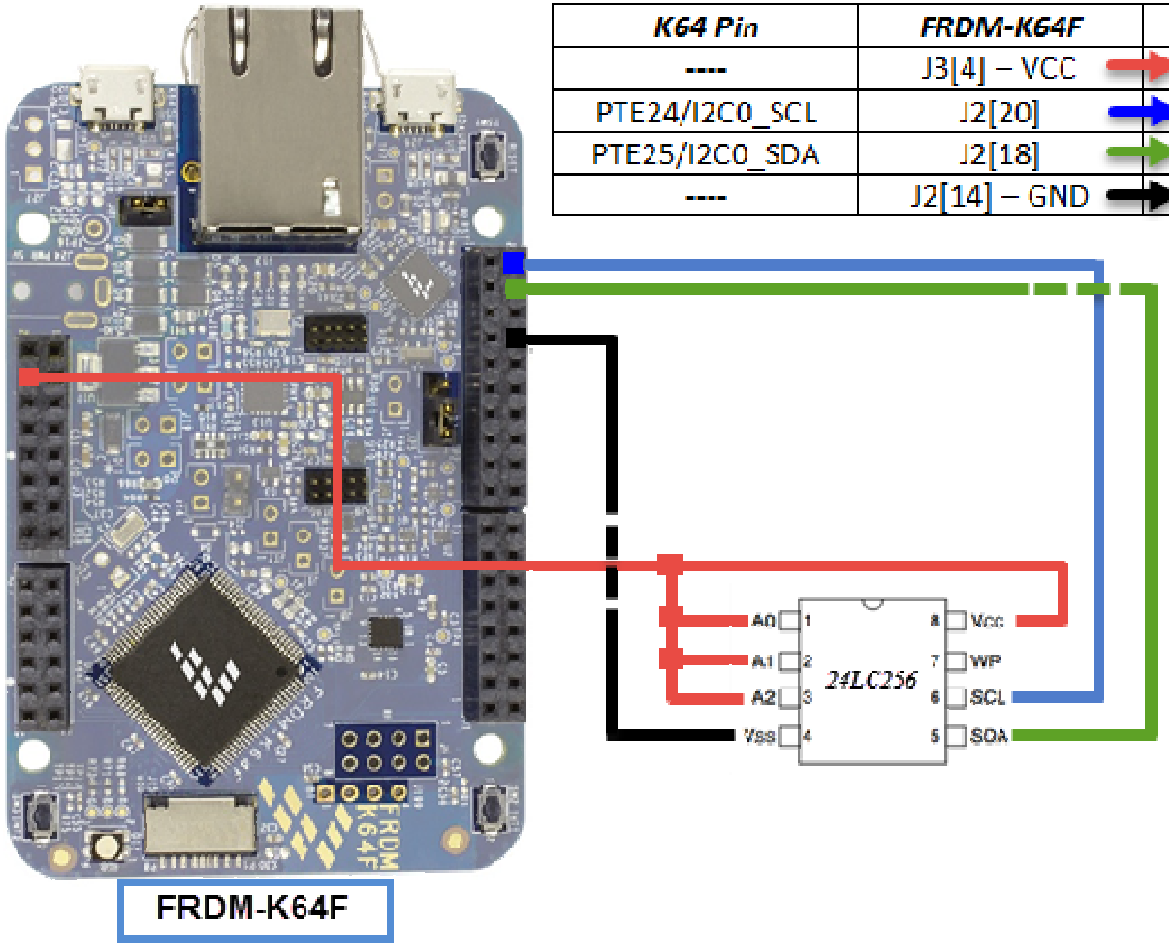
The Kinetis Software Development Kit includes an I2C driver to simplify the access to the I2C module of Kinetis devices. This simple driver allows establishing master – slave communications according to the I2C protocol.

The sample application

For testing the driver, an application was implemented to connect a FRDM-K64F with a 24LC256 EEPROM memory. The different functions used for testing the communication were: *Write Byte*, *Read Byte*, *Write Block* and *Read Block*.

***NOTE:** Before testing the example project it is necessary to have the KSDK library installed. This sample application was developed and tested with KSDK v1.0.0. It can be downloaded from www.freescale.com/ksdk. Once installed, the FRDM-K64F KSDK platform library has to be compiled. Only then it is possible to have a successful compilation of the example project.

- The connections to test this application are showed in the next figure:



K64 Pin	FRDM-K64F	I2C EEPROM
----	J3[4] – VCC	VCC, A0, A1, A2
PTE24/I2C0_SCL	J2[20]	SCL
PTE25/I2C0_SDA	J2[18]	SDA
----	J2[14] – GND	VSS

Connections between FRDM-K64F and 24LC256 EEPROM

The slave's data

A variable of type *i2c_device_t* must be created. *i2c_device_t* is a structure defined on the fsl_i2c_master_driver.h file, which can be found on *C:\Freescale\KSDK_1.0.0\platform\drivers\i2c\i2c_master*. Such variable defines the necessary information to establish communication with the slave; this information corresponds to the slave's address and the baud rate (kbps) of the slave device.

Example

```
i2c_device_t eeprom =                // Slave EEPROM settings
{
      .address = EEPROM_ADDRESS,
      .baudRate_kbps = BAUDRATE
};
```

I2C Master Driver APIs

I2C_DRV_MasterSendDataBlocking - This API is used in the implementation of Write functions.

```
i2c_status_t I2C_DRV_MasterSendDataBlocking(uint32_t instance,    // I2C peripheral instance number.
                                            const i2c_device_t * device, // Pointer to I2C slave information.
                                            uint8_t * cmdBuff,    // Pointer to the commands to be transferred.
                                            uint32_t cmdSize,     // Length in bytes of the commands to be transferred.
                                            uint8_t * txBuff,     // Pointer to the data to be sent.
                                            uint32_t txSize,      // Length in bytes of the data to be sent.
                                            uint32_t timeout_ms); // Timeout for the transfer (ms)
```

I2C_DRV_MasterReceiveDataBlocking - This API is used in the implementation of Read functions.

```
i2c_status_t I2C_DRV_MasterReceiveDataBlocking(uint32_t instance, // I2C peripheral instance number
                                               const i2c_device_t * device, // Pointer to I2C slave information.
                                               uint8_t * cmdBuff, // Pointer to the commands to be transferred.
                                               uint32_t cmdSize,  // Length in bytes of the commands to be transferred
                                               uint8_t * rxBuff,  // Pointer to the data to be received.
                                               uint32_t rxSize,   // Length in bytes of the data to be received.
                                               uint32_t timeout_ms); // Timeout for the transfer (ms)
```

Implemented functions

This section provides a brief description of the functions implemented to communicate with the I2C EEPROM memory:

- Write Byte
- Read Byte
- Write Block
- Read Block

Function "*Write Byte*"

```
void vfnWriteByte (uint16_t addressToWByte, uint8_t dataToWrite)
```

Parameters

`addressToWByte`	Address selected to write a byte.
`dataToWrite`	Data that will be written into the corresponding address.

Function "*Read Byte*"

```
uint8_t  bfnReadByte (uint16_t addressToRByte)
```

Parameters

`addressToRByte`	Address selected to read a byte.

Returns

This function returns a variable of type `uint8_t` containing the received data.

Function "*Write Block*"

```
void vfnWriteBlock (uint16_t addressToWBlock, uint8_t *dataToWriteBlock, uint16_t blockToWLength)
```

Parameters

`addressToWBlock`	Address selected to write a block.
`*dataToWriteBlock`	Pointer to the array containing the data that will be written.
`blockToWLength`	Length in bytes of the block that must be written.

Function "*Read Block*"

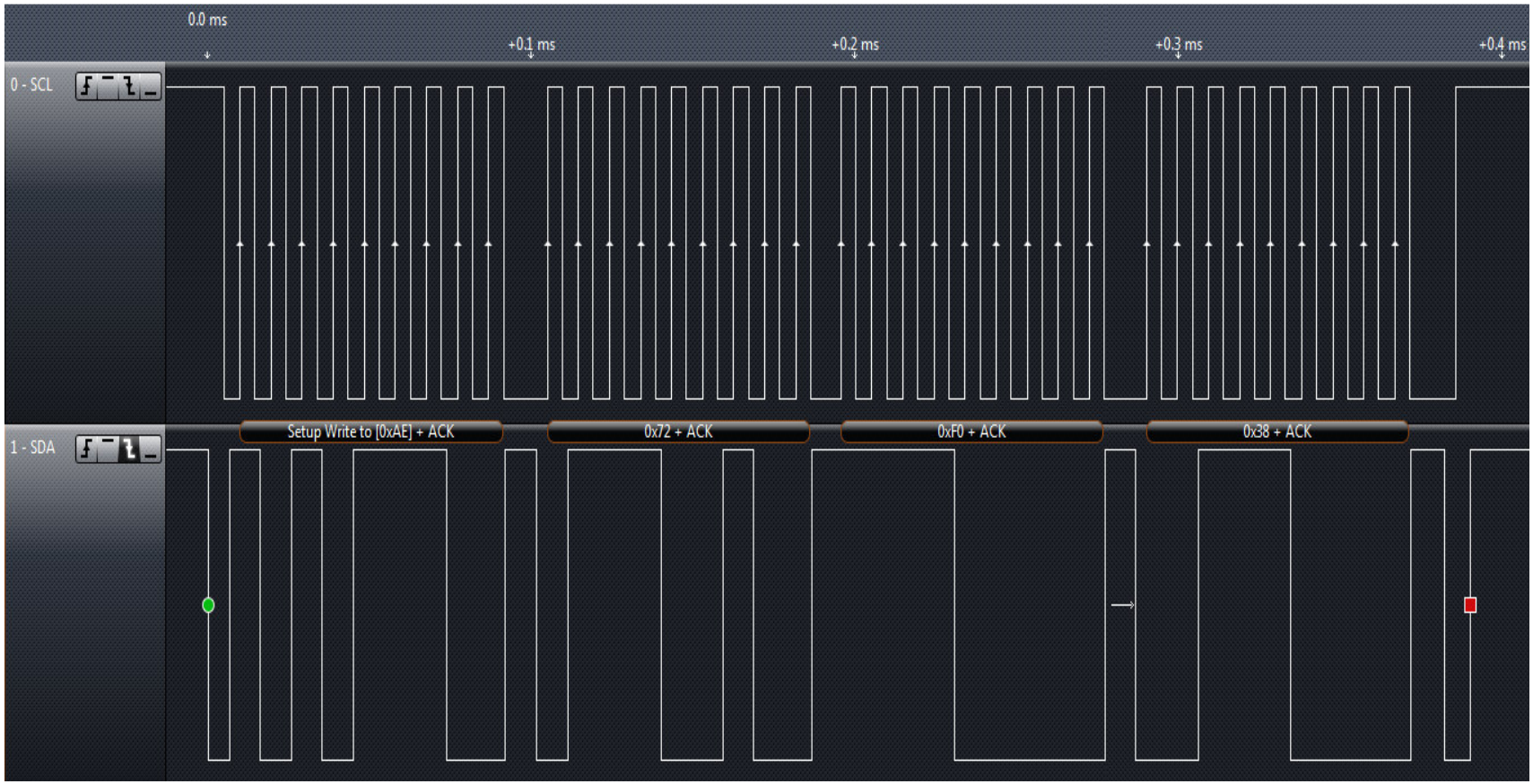
```
void vfnReadBlock (uint16_t addressToRBlock, uint8_t *receivedData, uint16_t blockToRLength)
```

Parameters

`addressToRBlock`	Address selected to read a block.
`*receivedData`	Pointer to the array which will store the block of data received.
`blockToRLength`	Length in bytes of the block that must be read.

Testing the functions

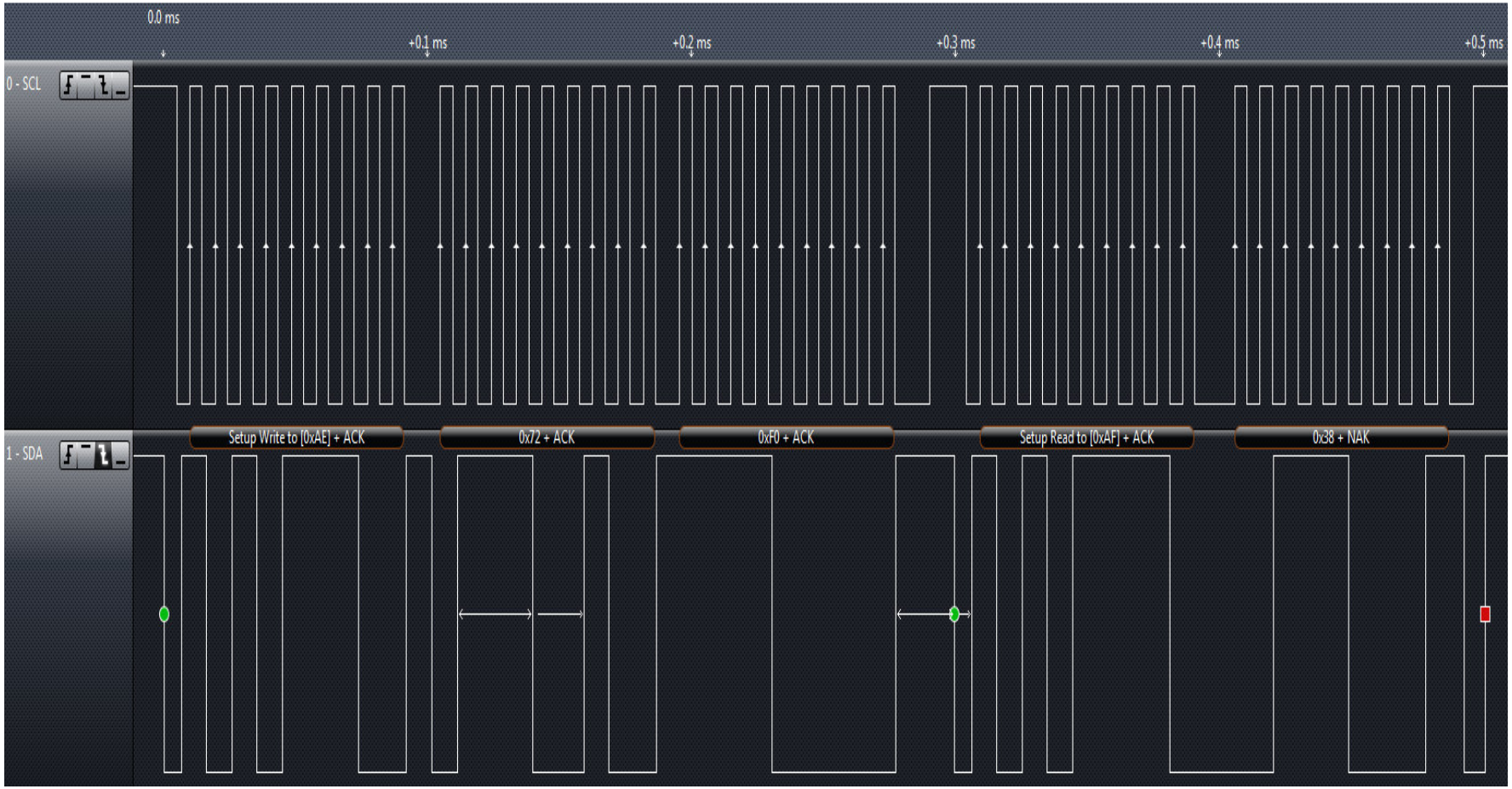
Function vfnWriteByte



Writing an 0x38 to the EEPROM memory address 0x72F0

The first sent frame corresponds to the EEPROM slave address concatenated with the R/W bit low (0xAE). The next two frames correspond to the EEPROM memory address to write (0x72F0). The last one is the corresponding data to be written (0x38).

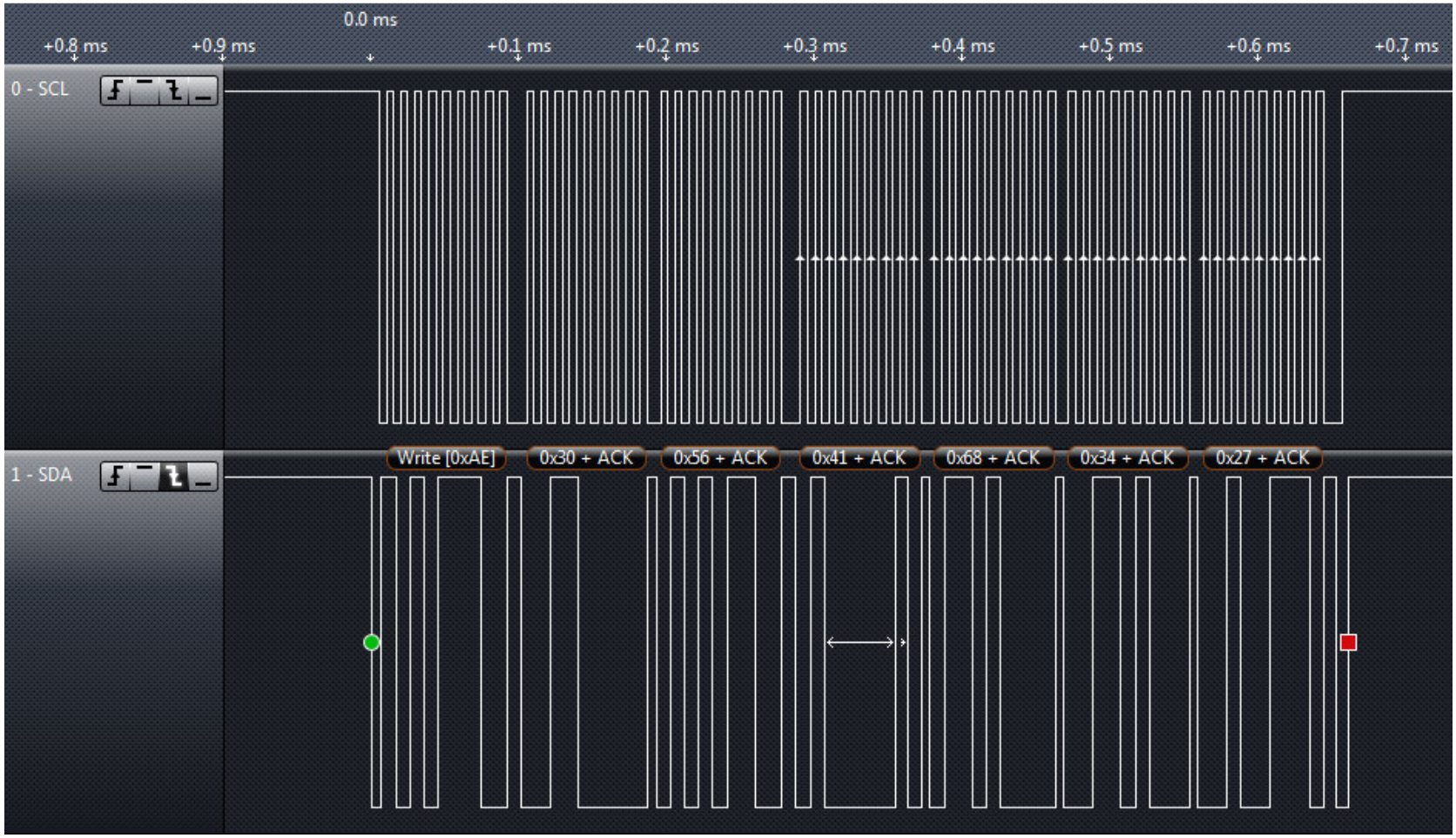
Function bfnReadByte



Reading the 0x38 stored at the EEPROM memory address 0x72F0

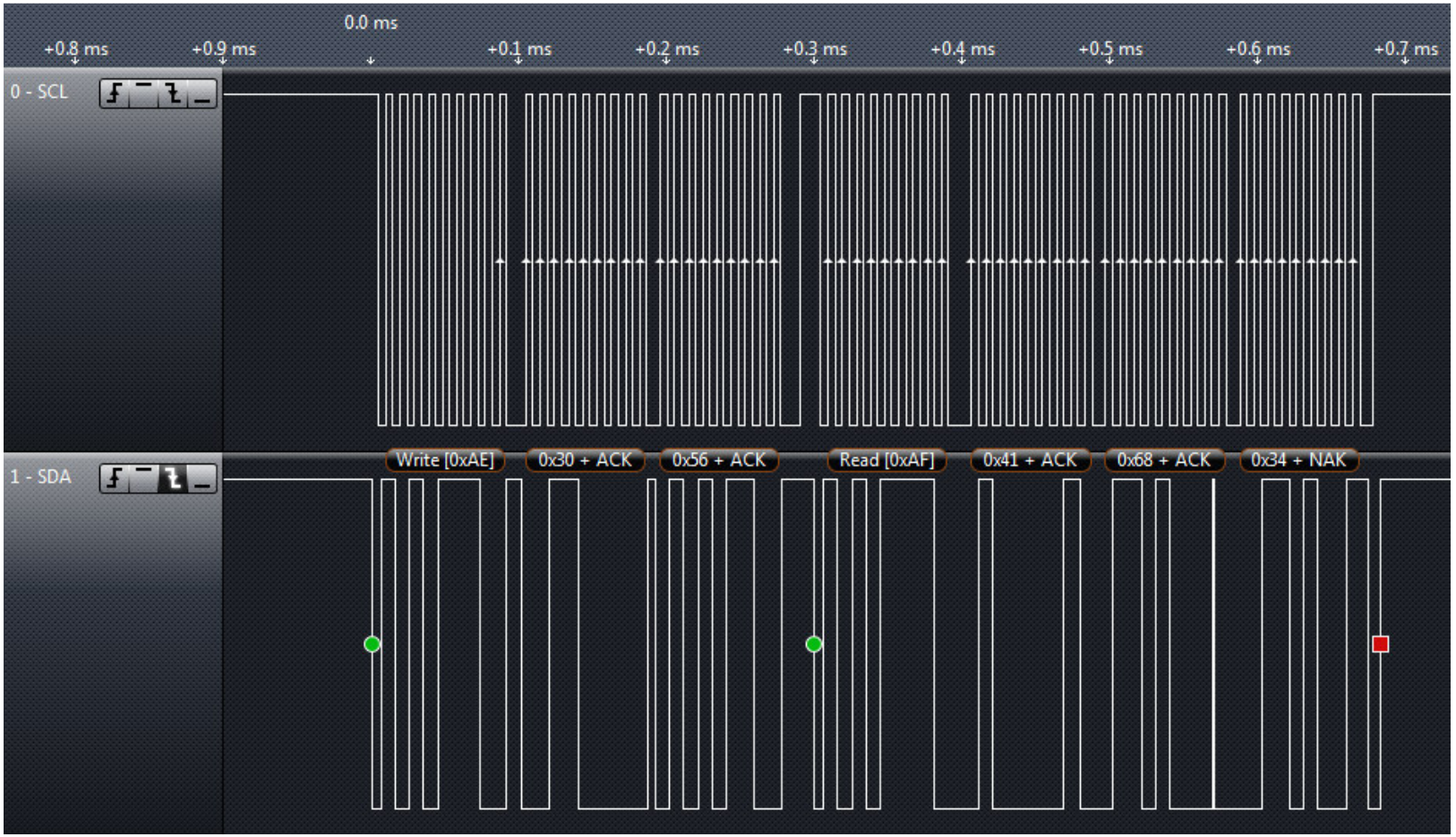
In this case, a Write command is first sent with R/W bit low (0xAE). Next are the two bytes corresponding to the EEPROM memory address to read (0x72F0). A Repeated Start is issued and then a Read command (0xAF) in order to receive the 0x38 byte stored at EEPROM memory address 0x72F0.

Function vfnWriteBlock



Writing a block of 4 bytes (0x41, 0x68, 0x34 and 0x27) to the EEPROM memory address 0x3056

Function vfnReadBlock



Reading a block of three bytes (0x41, 0x68 and 0x34) stored at the EEPROM memory address 0x3056