AD7745 / AD7746 SUPPORT DOCUMENT

MATTIA DEVANGELIO

Year 2019 - 2020

0.1 I2C COMMUNICATION

`Wire.beginTransmission(ADDRESS_CDC)` allows to begin a transmission to the slave device with the given address `ADDRESS_CDC` = 0x48, where the device address is specified in the datasheet.

Subsequently, bytes are queued for transmission with `Wire.write()` and transmitted by ending the transmission with the addressed device with `Wire.endTransmission()`.

Sequence:

```
1 Wire.beginTransmission(Address);
  Wire.write(Nbytes);
  Wire.endTransmission();
```

The following sequence allows to write a value on any slave's register:

```
  Wire.beginTransmission(Address);
2 Wire.write(Nbytes1);
  Wire.write(Nbytes2)
  Wire.endTransmission();
```

where `Nbytes1` is the slave's register which will be written on, and `Nbytes2` is the value to write on the register.

According to the mentioned sequence, the value of each register may be set by the `setRegister(r, val)` function: the value `val` is written on the called register `r`.

The `readfromregister(nbytes)` function allows to read the specified number of bytes `nbytes` from the registers sequentially, starting the reading process from the first addressed register, i. e. status register. By this feature, the byte of each register will be read one by one.

The function is based on `Wire.requestFrom(address, quantity)`, where the parameters are the device's address to request bytes from and the number of bytes to request:

```
1 Wire.requestFrom(ADDRESS_CDC, nbytes)
```

`Wire.read()` allows to read the byte transmitted from the slave device to the master after the `requestFrom()`.

In order to read sequentially `N` bytes, the content of `N` registers will be saved in a `N`-length array `d[]` dynamically.

The function `readcapdata` bases on the reported code to read sequentially from the three capacitance data registers:

```
unsigned char d[N];
char i=0;
setRegister(STATUS_REG,-1);
Wire.requestFrom(ADDRESS_CDC,N); //request N bytes from the slave
while(i<N)
{
while(!Wire.available()==N){
//waiting until the slave sends the requested bytes
}
d[i]=Wire.read(); //save the byte as character
i++;
}
```

Register	AP	Value
Capacitance Data H	0x01	H byte
Capacitance Data M	0x02	M byte
Capacitance Data L	0x03	L byte

Table 1

In the same way `readvtdata` reads sequentially from the three voltage data registers:

Register	AP	Value
Voltage Data H	0x04	H byte
Voltage Data M	0x05	M byte
Voltage Data L	0x06	L byte

Table 2

1. H byte ⟶ `d[0]`
2. M byte ⟶ `d[1]`
3. L byte ⟶ `d[2]`

```
unsigned long value=((unsigned long)d[0]<<16)+
                    ((unsigned long)d[1]<<8)+
                    (unsigned long)d[2];
```

0.1.1 *Cap set-up*

Bit set up:

- B0 = 0
  set to 0 for better conversion performance
- B1 = 0, B2 = 0, B3 = 0, B4 = 0
  default values for proper operation
- B5 = 0
  differential mode disabled → single mode
- B6 = 0
  second input channel disabled
- B7 = 1
  capacitive channel enabled

Register	AP	BIN	DEC	HEX
`CAP_SETUP_REG`	0x07	10000000	128	0x80

Table 3

```
setRegister(CAP_SETUP_REG, 128);
```

0.1.2 *Voltage set-up*

External reference voltage

Bit set up:

- B0 = 1
  set to 1 for better conversion performance
- B1 = 0
  short of the channel disabled
- B2 = 0, B3 = 0
  default values for proper operation
- B4 = 1
  external voltage reference on REFIN(+) REFIN(-)
- B5 = 1, B6 = 1
  external voltage input
- B7 = 1
  voltage channel enabled

Register	AP	BIN	DEC	HEX
`VT_SETUP_REG`	0x08	11110001	241	0xF1

Table 4

```
setRegister(VT_SETUP_REG, 241);
```

Internal reference voltage

Bit set up:

- B0 = 1
  set to 1 for better conversion performance
- B1 = 0
  short of the channel disabled
- B2 = 0, B3 = 0
  default values for proper operation
- B4 = 0
  external reference voltage disabled → 1.17 V internal reference voltage enabled

- B5 = 1, B6 = 1
  external voltage input
- B7 = 1
  voltage channel enabled

Register	AP	BIN	DEC	HEX
VT_SETUP_REG	0x08	11100001	225	0xE1

Table 5

```
setRegister(VT_SETUP_REG, 225);
```

0.1.3 *Exc set-up*

Bit set up:

- B0 = 1, B1 = 1
  excitation L level = 0, excitation H level = V_{dd}
- B2 = 0
  inverted excitation output A is disabled
- B3 = 1
  excitation output A is enabled
- B4 = 0
  inverted excitation output B is disabled
- B5 = 0
  excitation output B is disabled
- B6 = 1
  excitation signal present on the output during both capacitance and voltage conversion.
- B7 = 0
  set to 0 for better conversion performance

Register	AP	BIN	DEC	HEX
EXC_SETUP_REG	0x09	01001011	75	0x4B

Table 6

```
setRegister(EXC_SETUP_REG, 75);
```

0.1.4 *Configuration set-up*

Continuous conversion (fast mode)

Bit set up:

- B0 = 1, B1 = 0, B2 = 0
  mode: continuous conversion
- B3 = 0, B4 = 0, B5 = 0
  fastest conversion time on capacitive channel ($f_s = 90.9\,\text{Hz}$)
- B6 = 0, B7 = 0
  fastest conversion time on voltage channel ($f_s = 49.8\,\text{Hz}$)

Register	AP	BIN	DEC	HEX
`CONFIGURATION_REG`	0x0A	00000001	1	0x01

Table 7

```
setRegister(CONFIGURATION_REG, 1);
```

Continuous conversion (slow mode)

Bit set up:

- B0 = 1, B1 = 0, B2 = 0
  mode: continuous conversion
- B3 = 1, B4 = 1, B5 = 1
  slowest conversion time on capacitive channel ($f_s = 9.1\,\text{Hz}$)
- B6 = 1, B7 = 1
  slowest conversion time on voltage channel ($f_s = 8.2\,\text{Hz}$)

Register	AP	BIN	DEC	HEX
`CONFIGURATION_REG`	0x0A	11111001	249	0xF9

Table 8

```
setRegister(CONFIGURATION_REG, 249);
```

Capacitance offset calibration

Bit set up:

- B0 = 1, B1 = 0, B2 = 1
  mode: offset calibration
- B3 = 1, B4 = 1, B5 = 1
  slowest conversion time on capacitive channel ($f_s = 9.1\,\text{Hz}$)
- B6 = 1, B7 = 1
  slowest conversion time on voltage channel ($f_s = 8.2\,\text{Hz}$)

Register	AP	BIN	DEC	HEX
`CONFIGURATION_REG`	0x0A	11111101	253	0xFD

Table 9

```
setRegister(CONFIGURATION_REG, 253);
```

Capacitance/Voltage gain calibration

Bit set up:

- B0 = 0, B1 = 1, B2 = 1
  mode: gain calibration
- B3 = 1, B4 = 1, B5 = 1
  slowest conversion time on capacitive channel ($f_s = 9.1\,\text{Hz}$)
- B6 = 1, B7 = 1
  slowest conversion time on voltage channel ($f_s = 8.2\,\text{Hz}$)

Register	AP	BIN	DEC	HEX
`CONFIGURATION_REG`	0x0A	11111110	254	0xFE

Table 10

```
setRegister(CONFIGURATION_REG, 254);
```