Memoria EEPROM

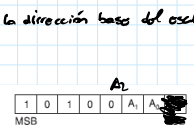
domingo, 20 de abril de 2025 1:18

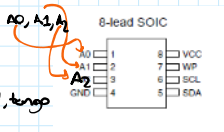
La memoria EEPROM que vamos a utilizar va a ser la AT24C256, datos importantes

- Internamente está organizada por 512 páginas de 64bytes cada una.
- Alimentación entre $2.7V < V < 5.5V$ [habrá que hacer pruebas si funciona igual de bien tanto con 3.3V como con 5V]
- Vamos a utilizar comunicación por I2C [hasta 1MHz modo fast]
- Podemos leer/escribir byte a byte o en páginas [64bytes]

Dirección del esclavo I2C:

La dirección base del esclavo es: 1010A2A1A0R/W → Estos bits dependen de los pines hardware A0, A1, A2





→ por lo tanto si no conecto estos pines/están conectados a gnd, tengo que la dirección del esclavo es:

1 0 1 0 0 0 0
A2 A1 A0

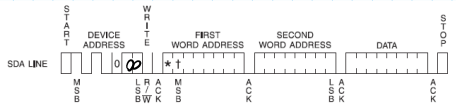
La dirección del esclavo es 0x50 pero: 0xA0 para escribir

0xA1 para leer

ESCRITURA EEPROM:

1. START
2. Enviar dirección esclavo [0x50] + "0" [escritura]
3. Enviar dirección alta [MSB]
4. Enviar dirección baja [LSB]
5. Enviar dato a escribir
6. STOP

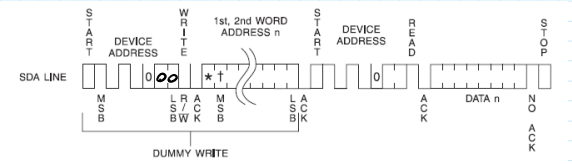
} Escritura de un byte [max 64bytes por página]



Cuando se hace una escritura la EEPROM necesita unos 5ms para grabar internamente.

LECTURA DESDE LA EEPROM:

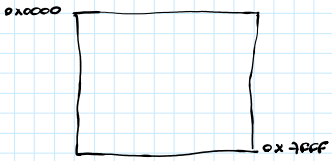
1. START
2. DIRRECCIÓN Esclavo + "0"
3. Enviar dirección alta
4. Enviar dirección baja
5. REPEAT START
6. Dirección esclavo + "1"
7. Recibes byte
8. Stop



Formula para celdas dentro de las páginas → Dirección = [nº de páginas] x [tamaño de página] + offset dentro de la página

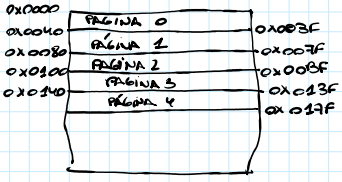
DATOS A GUARDAR EN LA MEMORIA

MEMORIA AT24C256



Plan A, guardar todo en una sola página [la 1ª] y que se vayan sobreescribiendo los datos

MEMORIA AT24C256



Plan B, guardar cada dato importante en una página diferente [guardar tantas veces se obtenga]

PAG 0 → temperatura [2bytes] | PAG 1 → humedad [2bytes] | PAG 2 → calidad del aire [2bytes] | PAG 3 → luminosidad [2bytes]

PAG 4 → hora y fecha que se han cogido las medidas [6bytes] | PAG 5 → X datos que se nos pueda ocurrir

FUNCIONES DE C

WriteMemory:

```
uint8_t writememory( uint8_t* WriteData, uint32_t longwrite){

  //Limpiamos los flags de eventos al realizar una nueva transmisión
  I2C_Event = 0U;
  //Transmitimos la dirección del registro donde escribir y sus datos
  I2Cdrv->MasterTransmit (ADDRESS_MEMORIA, WriteData, longwrite, false);
  //Esperemos hasta que se realice la transmisión
  while ((I2C_Event & ARM_I2C_EVENT_TRANSFER_DONE) == 0U);
  //Comprobamos que todos los datos se han transmitido
  if ((I2C_Event & ARM_I2C_EVENT_TRANSFER_INCOMPLETE) != 0U) return -1;

  return 0;
}
```

→ se inicia una transmisión en modo maestro, donde:

- ADRESS_memoria, es la dirección del esclavo [0x50]
- writedata, es la dirección interna [2bytes] + todos los datos que quiera escribir
- longwrite, es la longitud total del array que se van a transmitir por I2C.
- false, indica que no se hace un STOP automáticamente [por si hago más de una escritura]

ReadMemory

```
uint8_t readmemory(uint8_t* Registerdirectory, uint8_t* data, uint32_t longlecture){

  //Direccion del registro
    //Limpiamos los flags de eventos al realizar una nueva transmisión
  I2C_Event = 0U;
    //Transmitimos la dirección del registro a leer
  I2Cdrv->MasterTransmit (ADDRESS_MEMORIA, Registerdirectory, 2, true);
    //Esperemos hasta que se realice la transmisión
  while ((I2C_Event & ARM_I2C_EVENT_TRANSFER_DONE) == 0U);
    //Comprobamos que todos los datos se han transmitido
```

↑ puntero a los dos bytes de la dirección interna que quiero leer. ↑ longitud total de lo que quiero leer. ↑ datos leídos

→ con esto escribimos/dejamos el puntero en la sección interna donde voy a querer leer. Y esta a true, porque indica que se espera otro START [repeated start] no un STOP.

```
  //Limpiamos los flags de eventos al realizar una nueva transmisión
  I2C_Event = 0U;
    //Transmitimos la dirección del registro a leer
  I2Cdrv->MasterTransmit (ADDRESS_MEMORIA, Registerdirectory, 2, true);
    //Esperemos hasta que se realice la transmisión
  while ((I2C_Event & ARM_I2C_EVENT_TRANSFER_DONE) == 0U);
    //Comprobamos que todos los datos se han transmitido
  if ((I2C_Event & ARM_I2C_EVENT_TRANSFER_INCOMPLETE) != 0U) return -1;

  //Lectura datos registro
    //Limpiamos los flags de eventos al realizar una nueva transmisión
  I2C_Event = 0U;
    //Solicitamos recibir los datos
  I2Cdrv->MasterReceive (ADDRESS_MEMORIA, data, longlecture , false);
    //Esperemos hasta que se realice la transmisión
  while ((I2C_Event & ARM_I2C_EVENT_TRANSFER_DONE) == 0U);
    //Comprobamos que todos los datos se han transmitido
  if ((I2C_Event & ARM_I2C_EVENT_TRANSFER_INCOMPLETE) != 0U) return -1;

  return 0;
}
```

con esto escribimos/dejamos el puntero en la sección interna donde voy a querer leer.
Y esta a true, porque indica que se espera otro START [repeated start] no un STOP.

longlecture van a ser los bytes que quiero leer [desde donde esta el puntero] y en data se guardan todos estos bytes.
Como tiene argumento false, significa que se acaba con un STOP y se finaliza la transmisión.

función genérica para guardar medidas de dos bytes

```
void guardar_medida(uint16_t base_addr, uint8_t* data, uint8_t data_len, uint8_t* index) {
    // Calculamos cuántas entradas caben por página según el tamaño de los datos
    uint8_t max_entries = PAGE_SIZE / data_len;

    // Si el índice supera el límite, lo reseteamos
    if (*index >= max_entries) {
        *index = 0;
    }

    // Dirección en memoria para esta medida
    uint16_t addr = base_addr + (*index * data_len);

    // Preparar el buffer: 2 bytes de dirección + data_len bytes de dato
    write_buffer[0] = (addr >> 8) & 0xFF;  // Dirección MSB
    write_buffer[1] = addr & 0xFF;         // Dirección LSB

    for (uint8_t i = 0; i < data_len; i++) {
        write_buffer[2 + i] = data[i];     // Copiamos el dato al buffer
    }

    writememory(write_buffer, 2 + data_len);  // Dirección + dato

    (*index)++;  // Aumentamos el índice
}
```

tamaño de página / tamaño del dato = 64 / 2 = 32 bytes
(como siempre va a ser 2)

Esto lo utilizo para que si llega al final de la página, empiece a sobrescribir los datos mas antiguos.

base_addr es el comienzo de pagina que he definido anteriormente

PAG 0 → temperatura [2 bytes]
PAG 1 → humedad [2 bytes]
PAG 2 → calidad del aire [2 bytes]
PAG 3 → luminosidad [2 bytes]
PAG 4 → hora y fecha que se han cogido las medidas
PAG 5 → X datos que se nos pueda ocurrir

sabiendo por ejemplo que el comienzo de mi página para temperatura es 0x0000 entonces base_addr = 0x0000.
Y dependiendo de los datos que vamos escribiendo [index] vamos rellenando la página

función que voy a utilizar tanto para temperatura, humedad, luminosidad y calidad de aire.

```
void guardar_temperatura(uint16_t medida) {
    uint8_t data[2];
    data[0] = (medida >> 8) & 0xFF;
    data[1] = medida & 0xFF;

    guardar_medida(TEMP_PAGE_ADDR, data, 2, &temp_index);
}
```

medida que quiero guardar

guardo la medida en el buffer

utilizo la función genérica anterior

porque se de antemano que voy a escribir solo dos bytes de datos.