Driver ADS1299

Ing. Leandro Arrieta
Protocolos de Comunicación en Sistemas Embebidos
https://github.com/leaiava/driver_ADS1299

04/10/2021

ADS1299 Evaluation Board

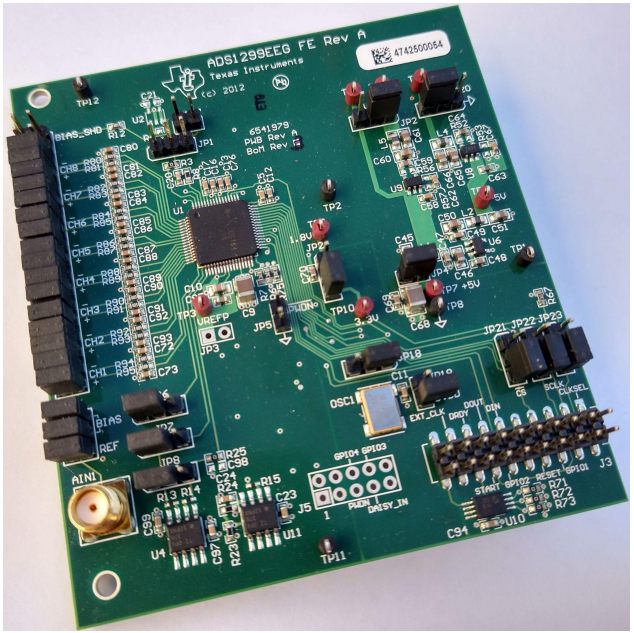
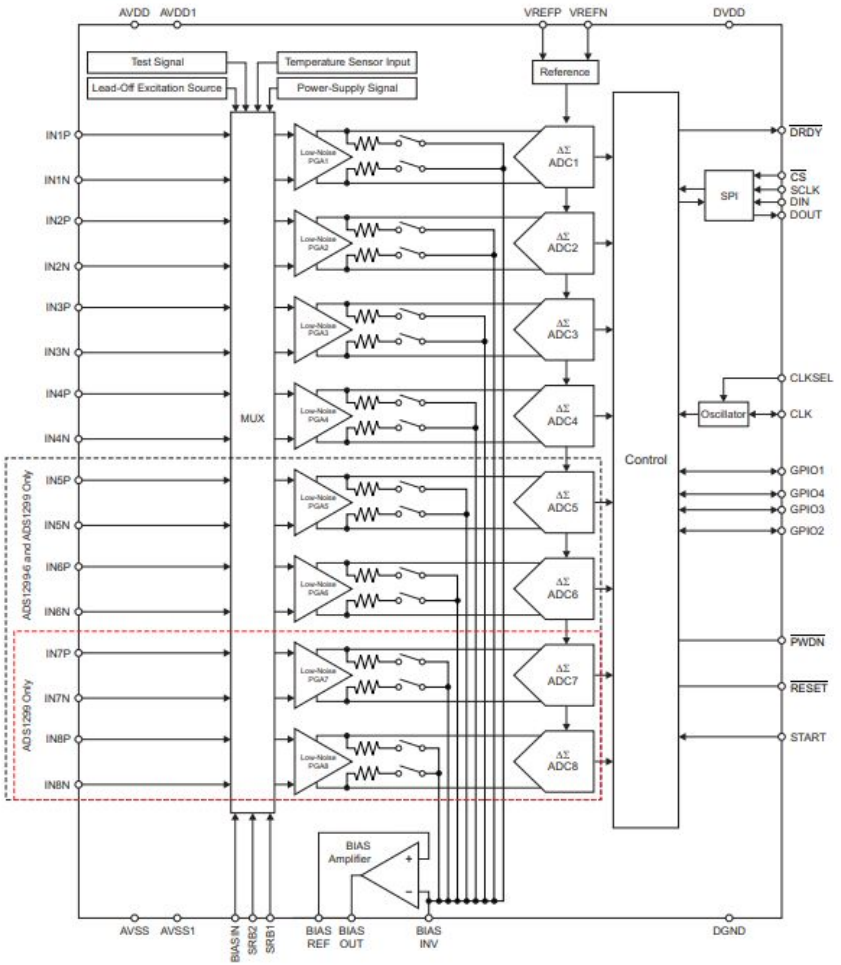
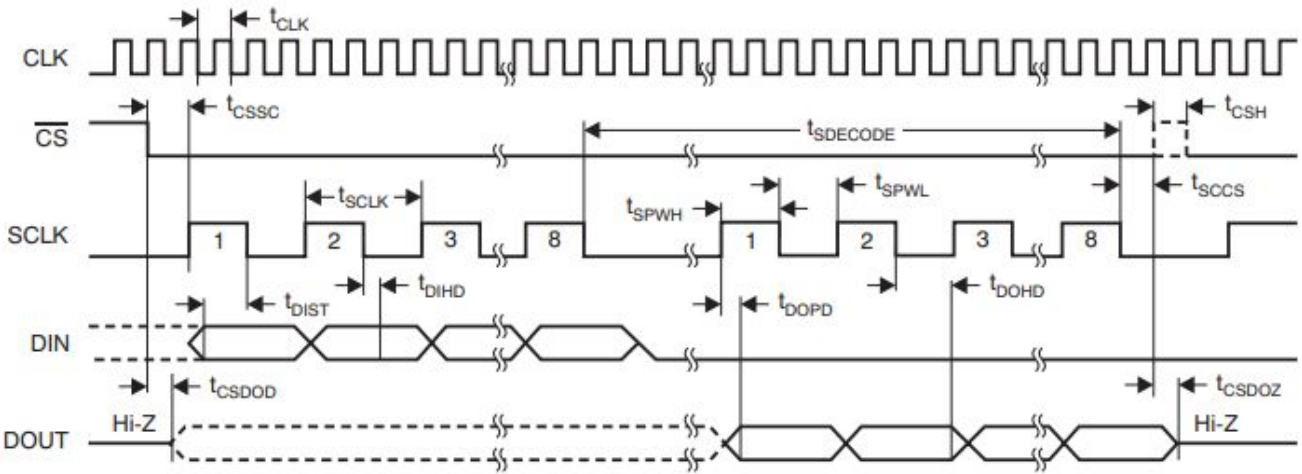


Diagrama en Bloques del ADS1299



Timings SPI ADS1299



NOTE: SPI settings are CPOL = 0 and CPHA = 1.

Comandos del ADS1299

COMMAND	DESCRIPTION	FIRST BYTE	SECOND BYTE
System Commands			
WAKEUP	Wake-up from standby mode	0000 0010 (02h)	
STANDBY	Enter standby mode	0000 0100 (04h)	
RESET	Reset the device	0000 0110 (06h)	
START	Start and restart (synchronize) conversions	0000 1000 (08h)	
STOP	Stop conversion	0000 1010 (0Ah)	
Data Read Commands			
RDATAC	Enable Read Data Continuous mode. This mode is the default mode at power-up.(1)	0001 0000 (10h)	
SDATAC	Stop Read Data Continuously mode	0001 0001 (11h)	
RDATA	Read data by command; supports multiple read back.	0001 0010 (12h)	
Register Read Commands			
RREG	Read *n nnnn* registers starting at address *r rrrr*	001*r rrrr* (2xh)(2)	000*n nnnn*(2)
WREG	Write *n nnnn* registers starting at address *r rrrr*	010*r rrrr* (4xh)(2)	000*n nnnn*(2)

Estructura del driver

Estructura con punteros a funciones para separar la capa más baja del driver

```
typedef struct {
    hardwareInit_t hardwareInit_fnc;
    csFunction_t chip_select_ctrl;
    spiWrite_t spi_write_fnc;
    spiRead_t spi_read_fnc;
    delayus_t delay_us_fnc;
    startFunction_t start_ctrl;     //!< Opcional
    resetFunction_t reset_ctrl;     //!< Opcional
    clkSelFunction_t clksel_ctrl;   //!< Opcional
    pwdnFunction_t pwdn_ctrl;       //!< Opcional
}ads1299_func_t;
```

Principales Funciones

```
void ads1299_writeRegister(registers_t registro, uint8_t data){
    ads1299_control.chip_select_ctrl(CS_ENABLE);
    ads1299_control.spi_write_fnc( SDATAC_CMD);
    ads1299_control.spi_write_fnc( WREG_CMD | registro );
    ads1299_control.spi_write_fnc( NULL_CMD);
    ads1299_control.spi_write_fnc( data );
    ads1299_control.delay_us_fnc(2);
    ads1299_control.chip_select_ctrl(CS_DISABLE);
}
```

Principales Funciones

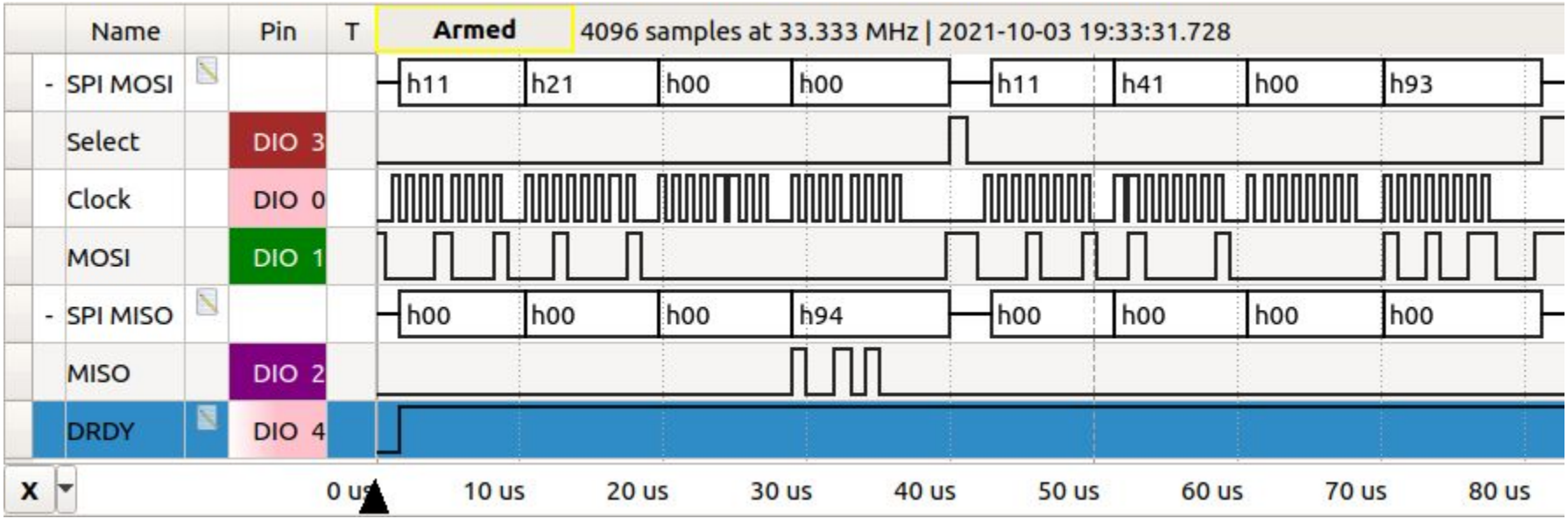
```
uint8_t ads1299_readRegister(registers_t registro){
    uint8_t readValue;
    ads1299_control.chip_select_ctrl(CS_ENABLE);
    ads1299_control.spi_write_fnc( SDATAC_CMD);
    ads1299_control.spi_write_fnc( RREG_CMD | registro );
    ads1299_control.spi_write_fnc( NULL_CMD);
    readValue = ads1299_control.spi_read_fnc();
    ads1299_control.delay_us_fnc(2);
    ads1299_control.chip_select_ctrl(CS_DISABLE);
    return readValue;
}
```

Función ads1299_setDataRate

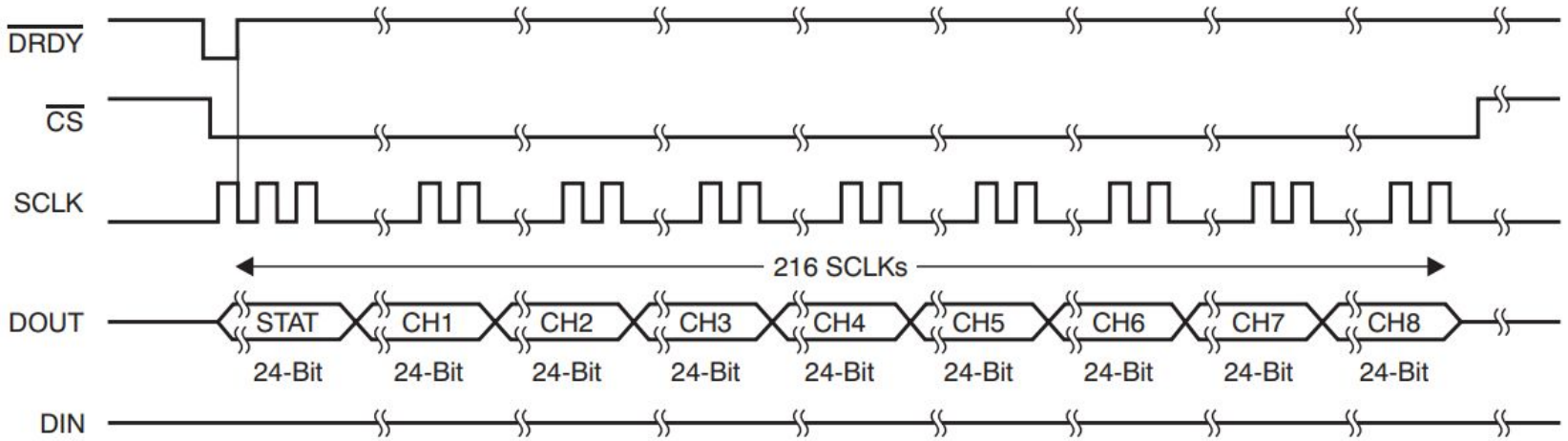
```
void ads1299_setDataRate(uint8_t config1Data){

    config1Data &= DATA_RATE_MASK;  // aplico mascara para solo tocar los bits del DATA_RATE
    uint8_t config1Reg = ads1299_readRegister(CONFIG1);
    config1Reg &= ~DATA_RATE_MASK;  // pongo en cero los bits del DATA_RATE
    config1Reg |= config1Data;      // aplico la configuracion del DATA_RATE
    ads1299_writeRegister(CONFIG1, config1Reg);
}
```

Captura con analizador lógico

ads1299_setDataRate(ADS1299_CONFIG1_DATA_RATE_500);



Data Output


DRDY
CS
SCLK
216 SCLKs
DOUT
STAT
CH1
CH2
CH3
CH4
CH5
CH6
CH7
CH8
24-Bit
24-Bit
24-Bit
24-Bit
24-Bit
24-Bit
24-Bit
24-Bit
24-Bit
DIN

Funciones ads1299_readData

```
void ads1299_readData(ads1299_data_t* ptrads1299_data){
    uint32_t data;
    ads1299_control.chip_select_ctrl(CS_ENABLE);
    for (uint8_t i=0 ; i < (CANTIDAD_DE_CANALES + 1) ; i++){
        data = 0;
        data = (uint32_t) ads1299_control.spi_read_fnc();
        data <<= 8;
        data |= (uint32_t) ads1299_control.spi_read_fnc();
        data <<= 8;
        data |= (uint32_t) ads1299_control.spi_read_fnc();
        ptrads1299_data->data[i] = (int32_t) data;
    }
    ads1299_control.delay_us_fnc(2);
    ads1299_control.chip_select_ctrl(CS_DISABLE);
}
```