**Id 32 bits unsigned**

**MCP2515.RESET()**

{

SPI.transfer(0xc0)

SPI.START()

SPI.transfer(0x02)

SPI.transfer(mcp\_TXB0CTRL) 0x30

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

EndSPI();

SPI.START()

SPI.transfer(0x02)

SPI.transfer(mcp\_TXB1CTRL) 0x40

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

EndSPI();

SPI.START()

SPI.transfer(0x02)

SPI.transfer(mcp\_TXB2CTRL) 0x50

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

SPI.transfer(0x00)

EndSPI();

SPI.START()

SPI.transfer(0x02)

SPI.transfer(MCP\_RXB0CTRL) 0x60

SPI.transfer(0x00)

EndSPI();

SPI.START()

SPI.transfer(0x02)

SPI.transfer(MCP\_RXB1CTRL) 0x70

SPI.transfer(0x00)

EndSPI();

SPI.START()

SPI.transfer(0x02)

SPI.transfer(MCP\_CANINTE) 0x2B

aa = CANINTF\_RX0IF | CANINTF\_RX1IF | CANINTF\_ERRIF | CANINTF\_MERRF

aa = 0x01 | 0x02 | 0x20 | 0x80 = A3

SPI.transfer(0xA3)

EndSPI();

SPI.START()

SPI.transfer(INSTRUCTION\_BITMOD) 0x05

SPI.transfer(MCP\_RXB0CTRL) 0x60

aa = RXBnCTRL\_RXM\_MASK | RXB0CTRL\_BUKT | RXB0CTRL\_FILHIT\_MASK | RXBnCTRL\_RXM\_STDEXT | RXB0CTRL\_BUKT | RXB0CTRL\_FILHIT

aa = 0x60 | 0x04 | 0x03 | 0x00 | 0x04 | 0x00 = 67

SPI.transfer(0x67)

EndSPI();

SPI.START()

SPI.transfer(INSTRUCTION\_BITMOD) 0x05

SPI.transfer(MCP\_RXB1CTRL) 0x70

aa = RXBnCTRL\_RXM\_MASK | RXB1CTRL\_FILHIT\_MASK | RXBnCTRL\_RXM\_STDEXT | RXB1CTRL\_FILHIT)

aa = 0x60 | 0x07 | 0x00 | 0x01 = 67

SPI.transfer(0x67)

EndSPI();

setFilter(filters[i], ext, 0)

setConfigMode();

setMode(CANCTRL\_REQOP\_CONFIG);

modifyRegister(MCP\_CANCTRL, CANCTRL\_REQOP,CANCTRL\_REQOP\_CONFIG);

SET FILTER UNA VEZ POR CADA MÁSCARA mask = RXF0, RXF1, RXF2, RXF3, RXF4, RXF5

bool ext = (i == 1);

StartSPI()

SPI.transfer(INSTRUCTION\_BITMOD) 0X05

SPI.transfer(MCP\_CANCTRL) 0X0F

SPI.transfer(CANCTRL\_REQOP) 0XE0

SPI.transfer(CANCTRL\_REQOP\_CONFIG) 0X80

END SPI

reg = MCP\_RXF0SIDH;

prepareId(uint8\_t \*buffer, const bool ext, const uint32\_t id)

uint16\_t canid = (uint16\_t)(id & 0x0FFFF);

if(ext)

{

buffer[MCP\_EID0] = (uint8\_t) (canid & 0xFF);

buffer[MCP\_EID8] = (uint8\_t) (canid >> 8);

canid = (uint16\_t)(id >> 16);

buffer[MCP\_SIDL] = (uint8\_t) (canid & 0x03);

buffer[MCP\_SIDL] += (uint8\_t) ((canid & 0x1C) << 3);

buffer[MCP\_SIDL] |= TXB\_EXIDE\_MASK;

buffer[MCP\_SIDH] = (uint8\_t) (canid >> 5);

}

Else{

buffer[MCP\_SIDH] = (uint8\_t) (canid >> 3);

buffer[MCP\_SIDL] = (uint8\_t) ((canid & 0x07 ) << 5);

buffer[MCP\_EID0] = 0;

buffer[MCP\_EID8] = 0;

}

setRegisters(reg, tbufdata, 4);

startSPI()

SPI.transfer(INSTRUCTION\_WRITE) 0x02

SPI.transfer(MCP\_RXF0SIDH)

FOR(INT I = 0; I < N; I++)

{

SPI.transfer(buffer[i]);

}

FIN SET FILTER;

MASK masks[] = {MASK0, MASK1};

for (int i=0; i<2; i++) {

setFilterMask(masks[i], true, 0);

}

setFilterMask

{

StartSPI();

SPI.transfer(INSTRUCCIÓN\_BITMOD) 0X05

SPI.transfer(mcp\_canctrl) 0x0F

SPI.transfer(canctrl\_reqop) 0XE0

SPI.transfer(CANCTRL\_REQOP\_CONFIG) 0x80

ENDSPI

}

}

**SETBITRATE**

**mcp2515.setBitrate(CAN\_125KBPS,MCP\_8MHZ);**

cfg1 = MCP\_8MHz\_125kBPS\_CFG1;

cfg2 = MCP\_8MHz\_125kBPS\_CFG2;

cfg3 = MCP\_8MHz\_125kBPS\_CFG3;

setRegister(MCP\_CNF1, cfg1); 0x2A 0x01

StartSPI();

SPI.transfer(INSTRUCTION\_WRITE) 0x02

SPI.transfer(MCP\_CNF1) 0x2A

SPI.transfer(cfg1) 0x01

ENDSPI()

setRegister(MCP\_CNF2, cfg2); 0x29 0XB1

StartSPI();

SPI.transfer(INSTRUCTION\_WRITE) 0x02

SPI.transfer(MCP\_CNF2) 0x2A

SPI.transfer(cfg2) 0x01

ENDSPI()

setRegister(MCP\_CNF3, cfg3); 0x28 0x85

StartSPI();

SPI.transfer(INSTRUCTION\_WRITE) 0x02

SPI.transfer(MCP\_CNF3) 0x2A

SPI.transfer(cfg3) 0x01

ENDSPI()

**FINSETBITRATE**

**setMode(CANCTRL\_REQOP\_NORMAL);**

modifyRegister(MCP\_CANCTRL, CANCTRL\_REQOP, CANCTRL\_REQOP\_NORMAL);

StartSPI();

SPI.transfer(INSTRUCTION\_BITMOD) 0x05

SPI.transfer(MCP\_CANCTRL) 0x0F

SPI.transfer(CANCTRL\_REQOP) 0Xe0

SPI.transfer(CANCTRL\_REQOP\_NORMAL) 0x00

ENDSPI;

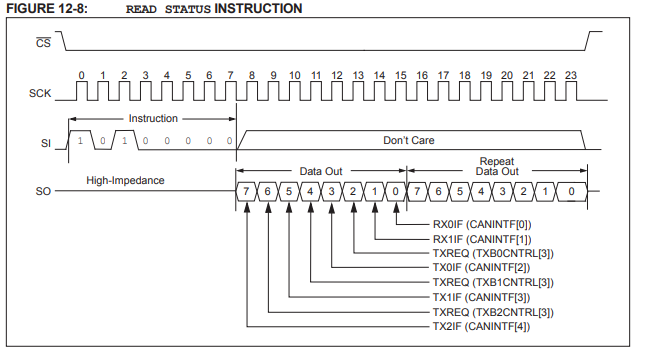
**MCP2515::readMessage(struct can\_frame \*frame)**

GetSTATUS()

STARTSPI()

SPI.TRANSFER(INSTRUCTION\_READ\_STATUS)

Rx\_data <= MISO\_byte

endSPI; 

if(Rx\_data & STAT\_RX0IF) then 0x01

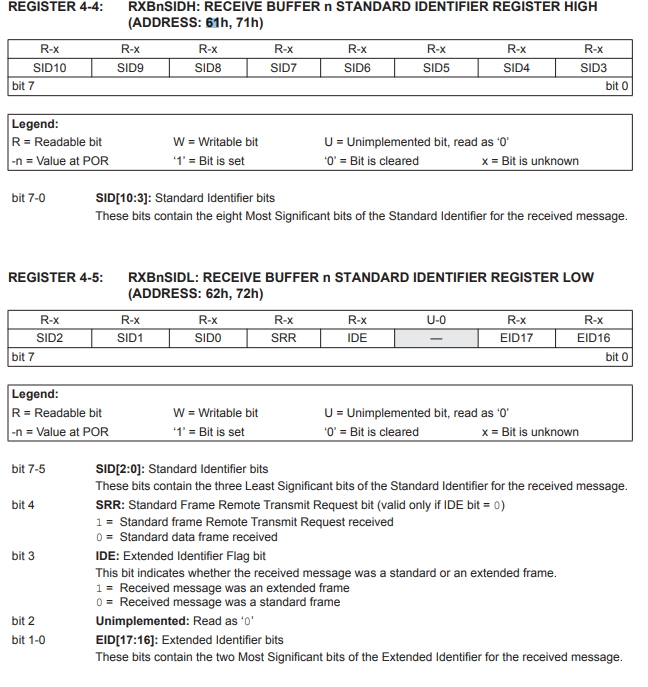
readMessage(RXB0, frame);

} else if ( stat & STAT\_RX1IF ) { 0x02

rc = readMessage(RXB1, frame);

end if;

READ MESSAGE



readRegisters(rxb->SIDH, tbufdata, 5);

STARTSPI()

SPI.transfer(Instruction\_READ) 0x03

SPI.transfer(SIDH) 0x61

For(int i = 0; i < 5 ; i++)

{

DATOS[I] = SPI.READ;

}

ID = (DATOS[MCP\_SIDH]<<3) + (DATOS[MCP\_SIDL]>>5);

if ( (tbufdata[MCP\_SIDL] & TXB\_EXIDE\_MASK) == TXB\_EXIDE\_MASK ) {

id = (id<<2) + (tbufdata[MCP\_SIDL] & 0x03);

id = (id<<8) + tbufdata[MCP\_EID8];

id = (id<<8) + tbufdata[MCP\_EID0];

id |= CAN\_EFF\_FLAG;

END IF;

dlc = (DATOS [MCP\_DLC] & DLC\_MASK); 0x0F

readRegister(rxb->CTRL)

STARTSPI()

SPI.transfer(Instruction\_READ) 0x03

SPI.transfer(CTRL) 0x61

if (ctrl & RXBnCTRL\_RTR) { 0x08 00001000 //si crtl tiene al menos un 1

id = id | CAN\_RTR\_FLAG; 0x40000000UL //or bit a bit

end if;

readRegisters(rxb->DATA, frame->data, dlc);

SPI.start()

SPI.transfer(InstructionREAD) 0x03

DATA[0] = SPI.transfer(0x66);

For(int i = 1 ; i < dlc; i++)

DATA[1] = SPI.transfer(0x66 + i)

END FOR

modifyRegister(MCP\_CANINTF, 0x03, 0); 0x20

CANINTF\_RX0IF = 0x01, CANINTF\_RX1IF = 0x02, => 0x03