Pseudo code 에서의 **I2C** 버스 드라이버

이것은 모든 프로그래머가 자신이 선호하는 언어로 이식(porting)할 수 있는 프로그래밍 언어인 PseudoCode 로 쓰여집니다.

우선 우리는 기초 인터페이스 루틴들의 세트를 정의할 것입니다. // 사이의 모든 텍스트는 remark 로 간주됩니다.

다음 변수들이 사용됩니다:

- n,x = 일반 목적의 BYTE
- SIZE = 한 번에 전송되는 데이터의 최대 수를 보유한 byte
- DATA(SIZE) = bytes 의 SIZE 수까지 유지하는 배열(array). 이것은 우리가 전송하고자 하는 데이터를 포함하며 수신한 데이터를 저장합니다.
- BUFFER = 즉시 수신된 또는 전송 데이터를 보유하는 byte 값.

```
/ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$/

/ *** I2C Driver V1.1 Written by V.Himpe. Released as Public Domain ***/

/ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$/

DECLARE N,SIZE,BUFFER,X Byte

DECLARE DATA() Array of SIZE elements

SUBroutine I2C_INIT / call this immediately after power-on /

    SDA=1

    SCK=0

    FOR n = 0 to 3

        CALL STOP

    NEXT n
```

```
ENDsub

SUBroutine START

        SCK=1 / BUGFIX !/

        SDA=1 / Improvement /

        SDA=0

        SCK=0

        SDA=1

ENDsub

SUBroutine STOP

        SDA=0

        SCK=1

        SDA=1

ENDsub

SUBroutine PUTBYTE(BUFFER)

        FOR n = 7 TO 0

                SDA= BIT(n) of BUFFER

                SCK=1

                SCK=0

        NEXT n

        SDA=1
```

```
ENDsub

SUBroutine GETBYTE

        FOR n = 7 to 0

                SCK=1

                BIT(n) OF BUFFER = SDA

                SCK=0

        NEXT n

        SDA=1

ENDsub

SUBroutine GIVEACK

        SDA=0

        SCK=1

        SCK=0

        SDA=1

ENDsub

SUBroutine GETACK

        SDA=1

        SCK=1

        WAITFOR SDA=0

        SCK=0
```

```
ENDSUB

/ this concludes the low-level set of instructions for the I2C driver

The next functions will handle the telegram formatting on a higher level /

SUBroutine READ(Device_address,Number_of_bytes)

    Device_adress=Device_adress OR (0000.0001)b /This sets the READ FLAG/

    CALL START

    CALL PUTBYTE(Device_adress)

    CALL GETACK

    FOR x = 0 to Number_of_bytes

        CALL GETBYTE DATA(x)=BUFFER /Copy received BYTE to DATA array /

        IF X< Number_of_bytes THEN /Not ack the last byte/

            CALL GIVEACK

        END IF

    NEXT x

    CALL STOP

ENDsub

SUBroutine WRITE(Device_address,Number_of_bytes)

    Device_adress=Device_adress AND (1111.1110)b / This clears READ flag /

    CALL START
```

```
        CALL PUTBYTE(Device_adress)
        CALL GETACK
        FOR x = 0 to Number_of_bytes
            CALL PUTBYTE (DATA(x))
            CALL GETACK
        NEXT x
        CALL STOP
ENDsub
SUBroutine RANDOMREAD(Device_adress,Start_adress,Number_of_bytes)
        Device_adress=Device_adress AND (1111.1110)b / This clears READ flag /
        CALL START
        CALL PUTBYTE(Device_adress)
        CALL GETACK
        CALL PUTBYTE(Start_adress)
        CALL GETACK
        CALL START /create a repeated start condition/
        Device_adress=Device_adress OR (0000.0001)b /This sets the READ FLAG/
        CALL PUTBYTE(Device_adress)
        CALL GETACK
```

```
        FOR x = 0 to Number_of_bytes
                CALL GETBYTE
                DATA(x)=BUFFER
                CALL GIVEACK
        NEXT x
        CALL STOP
ENDsub

SUBroutine RANDOMWRITE(Device_adress,Start_adress,Number_of_bytes)
        Device_adress=Device_adress AND (1111.1110)b / This clears READ
        flag /
        CALL START
        CALL PUTBYTE(Device_adress)
        CALL GETACK
        CALL PUTBYTE(Start_adress)
        CALL GETACK
        FOR x = 0 to Number_of_bytes
                CALL PUTBYTE (DATA(x))
                CALL GETACK
        NEXT x
        CALL STOP
ENDsub
```

```
/ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ /

/ **** I2C Driver . (c)95-97 V.Himpe . Public Domain release *** /

/ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ /
```

High level 루틴들에 관한 몇 가지 주의.

READ 와 WRITE 루틴은 한 개 또는 그 이상의 바이트를 슬레이브 디바이스로부터/슬레이브 디바이스에 read/write 합니다. 일반적으로 이것은 1 로 설정된 Number_of_bytes 하고만 사용됩니다. 예:

```
PCD8574=(0100.0000)b

CALL READ(PCD8574,1)

result = DATA(0)

/ will read the status of the 8 bit input port of a PCD8574. /

DATA(0)=(0110.01010)b

CALL WRITE(PCD8574,1)

/ will write 0110.0101 to the 8 bit port of the PCD8574 /
```

언제 multi-read 가 필요합니까? PCF8582 EEPROM 을 생각해 봅시다. 사용자는 한 번에 이것의 내용을 읽고자 합니다.

```
PCF8582=(1010.0000)b

CALL READ(PCF8582,255)
```

사용자는 동일한 것을 Number_of_bytes 가 사용자가 페이지 경계에 기록한 4 AND 보다 크지 않다는 제약하에서 EEPROM 의 WRITE 에서 할 수 있습니다 (오프셋을 위한 4 의 배수).

사용자는 부품 데이터쉬트를 확인해 보아야만 할 것입니다.

가장 유용한 명령은 RANDOMREAD 와 RANDOMWRITE 입니다.

EEPROM 의 20h 에 4 바이트의 데이터를 기록:

DATA(0)=(1010.0011)b

DATA(1)=(1110.0000)b

DATA(2)=(0000.1100)b

DATA(3)=(1111.0000)b

CALL RANDOMWRITE (PCF8582,(20)h,3)

동일한 것이 어드레스 42h 에서 시작하는 EEPROM 으로부터 16 바이트를 읽는 것에도 해당됩니다:

CALL RANDOMREAD(PCF8582,(42)h,15)

결과는 DATA 에 저장됩니다. 사용자가 할 일은 처리를 위해 배열에서 그것들을 읽는 것뿐입니다. 사용자가 디바이스 어드레스를 이러한 루틴들에 부여할 때는 R/W 플래그에 관해 주의할 필요가 없습니다. 이것은 자동으로 루틴 안에서 올바른 상태로 설정됩니다.