

COMUNICACIÓN C.D.C.

Ing. Juan Manuel Molina Amaro.
Depto. de
Instrumentación CICY Agosto 2011



CDC (Comunicación
Device Class).

**ES UNA CLASE DE COMUNICACIÓN DEL PROTOCOLO USB
DONDE PODEMOS INTERACTUAR CON UN PUERTO SERIE
VIRTUAL DEL TIPO RS232. MEDIANTE UN DRIVER
PROPORCIONADO POR MICROCHIP.**

CDC (Comunicación Device Class).

Es una comunicación a baja velocidad, en la cual el microcontrolador emula un puerto serial , utilizando un driver que provee directamente Microchip; la gran ventaja del protocolo es que podemos convertir una complicada comunicación USB, en una sencilla comunicación por RS232, que es la más común y que todos han podido utilizar.

Para la realización del firmware del micro controlador se utilizan las librerías
· `usb_desc_cdc.h` · `usb_cdc.h`

De las cuales nos interesa modificar la `usb_desc_cdc.h`, para personalizar valores del dispositivo, tales como el vendor id y product id (vid, pid), como se mostrará mas adelante.

Comunicación con otros dispositivos por medio de RS232
¡ACTUALMENTE NO DISPONIBLE EN LOS ORDENADORES!



Principalmente los ordenadores ya no cuentan con RS232
pero si con **¡USB!**

Ingredientes CDC:

	*	
--	---	--

firmware



LIBRERIAS

UBICACION

USB_CDC.H

C:\ARCHIVOS DE PROGRAMA\PICC\DRIVERS

USB_DESC_CDC.H

C:\ARCHIVOS DE PROGRAMA\PICC\DRIVERS

IDENTIFICADOR DE FABRICANTE

IDENTIFICADOR DEL PRODUCTO

firmware



La segunda fila es de vital importancia. Ya que este nos permite personalizar nuestro COM virtual VID: es un número de 16 bits que significa Vendor Identification o código que identifica al fabricante del hardware. EL Código de Microchip es 04D8h. Un VID = 04D8h.identifica a Microchip.

PID (USB Physical Interface Device): es un número de 16 bits que significa Product Identification o código que identifica al dispositivo. En éste caso lo cambiamos por 0044h.

Para poder establecer una comunicación USB se debe garantizar una frecuencia para el módulo USB de 48Mhz.

FUNCIONES QUE NOS PROPORCIONA LA LIBRERÍA USB_CDC.H

FUNCION	DESCRIPCION
USB_CDC_INIT()	DETERMINA LOS PARAMATROS DEL PUERTO CDC
USB_INIT()	CREA EL PUERTO VIRTUAL E INICIALIZA EL STACK
USB_INIT_CS()	CREA EL PUERTO VIRTUAL E INICIALIZA EL STACK
USB_ENUMERATE()	ESPERA HASTA QUE SEA ENUMERADO POR EL HOST
USB_CDC_KBHIT()	DETERMINA SI SE HA RECIBIDO UN CARÁCTER
USB_CDC_GETC()	OBTIENE EL CARÁCTER RECIBIDO EN EL BUFER RX
USB_CDC_PUTC()	PONE UN CARÁCTER EN EL BUFER DE TX
USB_TASK()	PERMITE CONECTAR AL BUS USB
USB_DETACH()	PERMITE DESCONECTAR DEL BUS

USB_CDC_INIT() CONFIGURACION DEL COM

```
void usb_cdc_init(void)
{
    usb_cdc_line_coding.dwDTERate = 9600;
    usb_cdc_line_coding.bCharFormat = 0;
    usb_cdc_line_coding.bParityType = 0;
    usb_cdc_line_coding.bDataBits = 8;
    (int8)usb_cdc_carrier = 0;
    usb_cdc_got_set_line_coding = FALSE;
    usb_cdc_break = 0;
    usb_cdc_put_buffer_nextin = 0;
    usb_cdc_get_buffer_status.got = 0;
}
```

USB_INIT()

ESTA FUNCION NOS CREA EL PUERTO VIRTUAL EN LA PC Y PONE A DISPOSICION UN STACK DE FUNCIONES PARA CONTROLAR LA COMUNICACIÓN CDC. Y ESPERA EN UN BUCLE INFINITO HASTA QUE EL DISPOSITIVO SEA CONECTADO AL BUS; PERO NO DETERMINA SI EL DISPOSITIVO HA SIDO ENUMERADO

USB_INIT_CS()

ESTA FUNCION HACE LO MISMO QUE USB_INIT(). PERO ESTA NO ESPERA EN UN CICLO INIFINITO A QUE EL DISPOSITIVO SEA CONECTADO.

USB_ENUMERATE()

ESTA FUNCION DEVUELVE VERDADERO SI EL DISPOSITIVO HA SIDO ENUMERDO POR LA PC.

USB_KBHIT()

ESTA FUNCION DEVUELVE VERDADERO SI HA RECIBIDO UN CARÁCTER EN EL BUFER DE RECEPCION.

USB_CDC_GETC() USB_CDC_PUTC()

ESTA FUNCION ES LEE O PONE EL CARÁCTER EN EL BUFER RX Y TX. PODRÍAN QUEDARSE INDEFINIDAMENTE ESPERANDO EL CARÁCTER. ESPERANDO SI EL BUFER ESTA LLENO.

PERSONALIZAR DESCRIPTOR

CREAR UNA CARPETA PRACTICAS CDC


COPIAR DESDE : \ARCHIVOS DE PROGRAMA \PICC \DRIVER

LOS ARCHIVOS **USB_CDC.H Y **USB_DESC_CDC.H** A
LA CARPETA CREADA**

DESCRIPTOR

USB_DESC_CDC.H

```
25  /// Changed device to USB 1.10          ///
26  ///                                     ///
27  //////////////////////////////////////////
28  ///      (C) Copyright 1996,2005 Custom Computer Services      ///
29  /// This source code may only be used by licensed users of the CCS  ///
30  /// C compiler. This source code may only be distributed to other  ///
31  /// licensed users of the CCS C compiler. No other use,          ///
32  /// reproduction or distribution is permitted without written    ///
33  /// permission. Derivative programs created using this software   ///
34  /// in object code form are not restricted in any way.           ///
35  //////////////////////////////////////////
36
37  #ifndef __USB_DESCRIPTOR__
38  #define __USB_DESCRIPTOR__
39
40  ////////////// config options, although it's best to leave alone for this demo //////////
41  #define USB_CONFIG_PID    0x0033
42  #define USB_CONFIG_VID    0x0461
43  #define USB_CONFIG_BUS_POWER 100 //100mA (range is 0..500)
44  #define USB_CONFIG_VERSION 0x0100 //01.00 //range is 00.00 to 99.99
45  ////////////// end config //////////////////////////////////////////
46
47  #define USB_HID_DEVICE FALSE
48  #define USB_CDC_DEVICE TRUE
49
50  #define USB_CDC_COMM_IN_ENDPOINT    1
51  #define USB_CDC_COMM_IN_SIZE        8
52  #define USB_EP1_TX_ENABLE USB_ENABLE_INTERRUPT
53  #define USB_EP1_TX_SIZE USB_CDC_COMM_IN_SIZE
```

**CAMBIAR
VID POR 0473
Y EL PID POR 0044**

**NOS DESPLAZAMOS HASTA
USB_DESC_STRING_TYPE**

CDC DESCRIPTOR

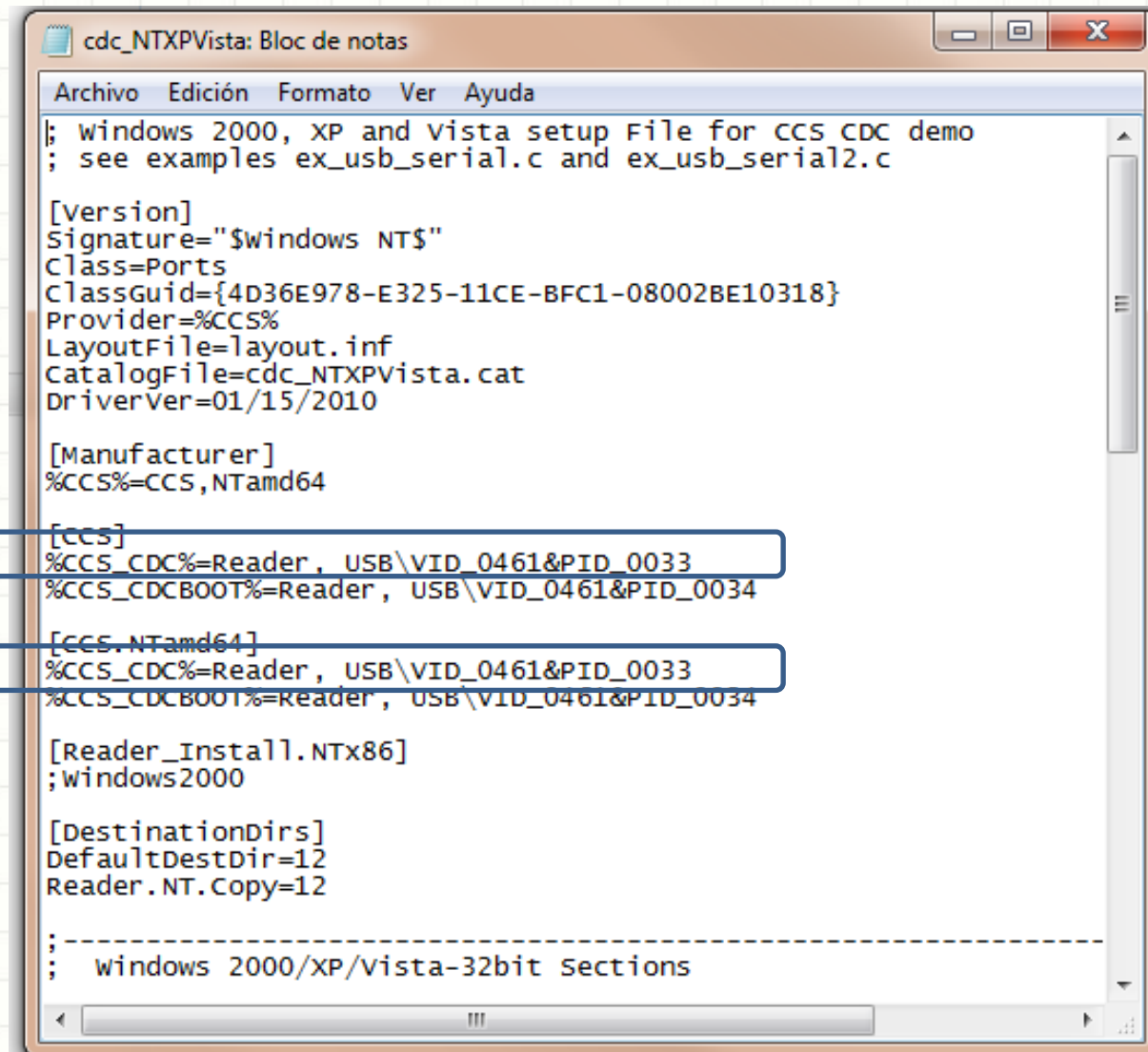
```
262         38, //length of string index
263         USB_DESC_STRING_TYPE, //descriptor type 0x03 (STRING)
264         'C',0,
265         'D',0,
266         'C',0,
267         ',',0,
268         'C',0,
269         'T',0,
270         'C',0,
271         'Y',0,
272         ',',0,
273         'D',0,
274         'T',0,
275         'P',0,
276         'L',0,
277         'O',0,
278         'M',0,
279         'A',0,
280         'D',0,
281         'O',0
282     };
283 #endif //!defined(USB_STRINGS_OVERWRITTEN)
```

Nombre que aparece en el modelo de la pestaña de propiedades del dispositivo

La librería USB_CDC .h hace referencia al descriptor

```
187
188 #if __USB_PIC_PERIF__
189 #if defined(__PCH__)
190 #include <pic18_usb.h> //Microchip 18Fxx5x hardware layer for usb.c
191 #else
192 #include <pic24_usb.h> //Microchip 18Fxx5x hardware layer for usb.c
193 #endif
194 #else
195 #include <usbn960x.h>
196 #endif
197 #include "usb_practica_desc_cdc.h" //USB Configuration and Device descriptors for this UBS device
198 #include <usb.c> //handles usb setup tokens and get descriptor reports
199
200 struct {
201     unsigned int32 dwDTERate; //data terminal rate, in bits per second
202     unsigned int8 bCharFormat; //num of stop bits (0=1, 1=1.5, 2=2)
203     unsigned int8 bParityType; //parity (0=none, 1=odd, 2=even, 3=mark, 4=space)
204     unsigned int8 bDataBits; //data bits (5,6,7,8 or 16)
205 } attribute ((packed)) usb_cdc_line_coding;
```

Personalización del Driver para windows



```
cdc_NTXPVista: Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda
; windows 2000, XP and vista setup File for CCS CDC demo
; see examples ex_usb_serial.c and ex_usb_serial2.c

[version]
Signature="$windows NT$"
Class=Ports
ClassGuid={4D36E978-E325-11CE-BFC1-08002BE10318}
Provider=%CCS%
LayoutFile=layout.inf
CatalogFile=cdc_NTXPVista.cat
DriverVer=01/15/2010

[Manufacturer]
%CCS%=CCS,NTamd64

[CCS]
%CCS_CDC%=Reader, USB\VID_0461&PID_0033
%CCS_CDCBOOT%=Reader, USB\VID_0461&PID_0034

[CCS.NTamd64]
%CCS_CDC%=Reader, USB\VID_0461&PID_0033
%CCS_CDCBOOT%=Reader, USB\VID_0461&PID_0034

[Reader_Install.NTx86]
;windows2000

[DestinationDirs]
DefaultDestDir=12
Reader.NT.Copy=12

;-----
; windows 2000/XP/Vista-32bit sections
```

Cambiar al valor
Requerido

Personalización del Driver para windows

Cambiar el texto
entre comillas por
el texto deseado

```
CDC_NTXPVista: Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda
ServiceBinary = %12%\usbser.sys
LoadOrderGroup = Base

;-----
; Vista-64bit Sections
;-----
[Reader.NTamd64]
Include=mdmcpq.inf
CopyFiles=FakeModemCopyFileSection
AddReg=Reader.NT.AddReg

[Reader.NTamd64.AddReg]
HKR,,DevLoader,,*ntkern
HKR,,NTMPDriver,,usbser.sys
HKR,,EnumPropPages32,, "MsPorts.dll,SerialPortPropPageProvider"

[Reader.NTamd64.Services]
AddService = usbser, 0x00000002, service_Instance.NTamd64

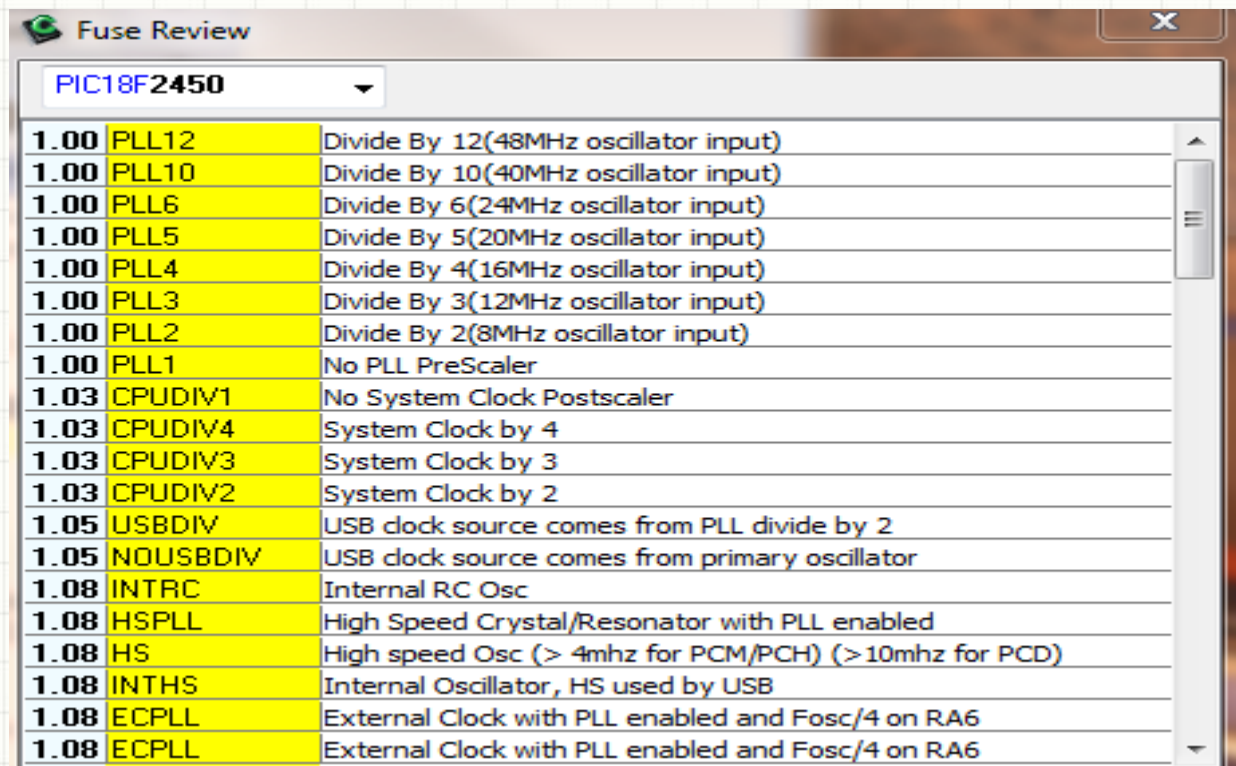
[Service_Instance.NTamd64]
DisplayName = %Serial.SvcDesc%
ServiceType = 1 ; SERVICE_KERNEL_DRIVER
StartType = 3 ; SERVICE_DEMAND_START
ErrorControl = 1 ; SERVICE_ERROR_NORMAL
ServiceBinary = %12%\usbser.sys
LoadOrderGroup = Base

[Strings]
CCS = "CICY Computer Services, Inc."
CCS_CDC = "USB PRACTICA DIPLOMADO CICY"
CCS_CDCBOOT = "CCS CDC Bootloader"
Serial.SvcDesc = "USB Serial emulation driver"
```


Preparando al pic para cdc

Paso 1 elegir el Micro `#include<18F2550>`

Paso 2 checar el PLL de los fuses adecuado para el cristal usado



PIC18F2450		
1.00	PLL12	Divide By 12(48MHz oscillator input)
1.00	PLL10	Divide By 10(40MHz oscillator input)
1.00	PLL6	Divide By 6(24MHz oscillator input)
1.00	PLL5	Divide By 5(20MHz oscillator input)
1.00	PLL4	Divide By 4(16MHz oscillator input)
1.00	PLL3	Divide By 3(12MHz oscillator input)
1.00	PLL2	Divide By 2(8MHz oscillator input)
1.00	PLL1	No PLL PreScaler
1.03	CPUDIV1	No System Clock Postscaler
1.03	CPUDIV4	System Clock by 4
1.03	CPUDIV3	System Clock by 3
1.03	CPUDIV2	System Clock by 2
1.05	USBDIV	USB clock source comes from PLL divide by 2
1.05	NOUSBDIV	USB clock source comes from primary oscillator
1.08	INTRC	Internal RC Osc
1.08	HSPLL	High Speed Crystal/Resonator with PLL enabled
1.08	HS	High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
1.08	INTHS	Internal Oscillator, HS used by USB
1.08	ECPLL	External Clock with PLL enabled and Fosc/4 on RA6
1.08	ECPLL	External Clock with PLL enabled and Fosc/4 on RA6

Fusibles que necesita la comunicación USB

HSPLL necesario cuando el cristal es mayor de 4MHZ

USBDIV Significa que el Clock del USB se tomará del divisor por 2 del PLL de 96MHz,

PLL n se define de acuerdo al valor del cristal usado

VREGEN Habilita el regulador de 3.3 volts que usa el Transceiver interno para el módulo USB.

Paso 3 incluimos las librerías que se requieren

Paso 4 Implementamos el programa

programa1



PROGRAMA 2



PROGRAMA 3



PROGRAMA 4

USB_BOOTLOADER

EL BOOTLOADER ES UNA PEQUEÑA APLICACIÓN QUE SE GRABA POR PRIMERA VEZ AL MICRO POR MEDIO DE UN PROGRAMADOR COMO EL “PICKIT2”, Y QUE POR MEDIO DE UNA PUSHBOTON SELECCIONA O NO EL BOOTLOADER

POSTERIORMENTE LA PROGRAMACION DE APLICACIONES SE HACE FACIL Y VELOZ POR MEDIO DEL PUERTO USB.

UBICACIÓN DEL BOOTLOADER

EL BOOTLOADER SE ENCUENTRA EN LAS PRIMERAS LOCALIDADES DE MEMORIA .

Interrupt vector remapping: Es una pequeña área de memoria de programa en donde están mapeados los vectores de reset, e interrupciones de alta y baja prioridad. Aquí es donde las aplicaciones subidas por el bootloader ubican los saltos a sus funciones de main

FUNCIONAMIENTO DEL BOOTLOADER

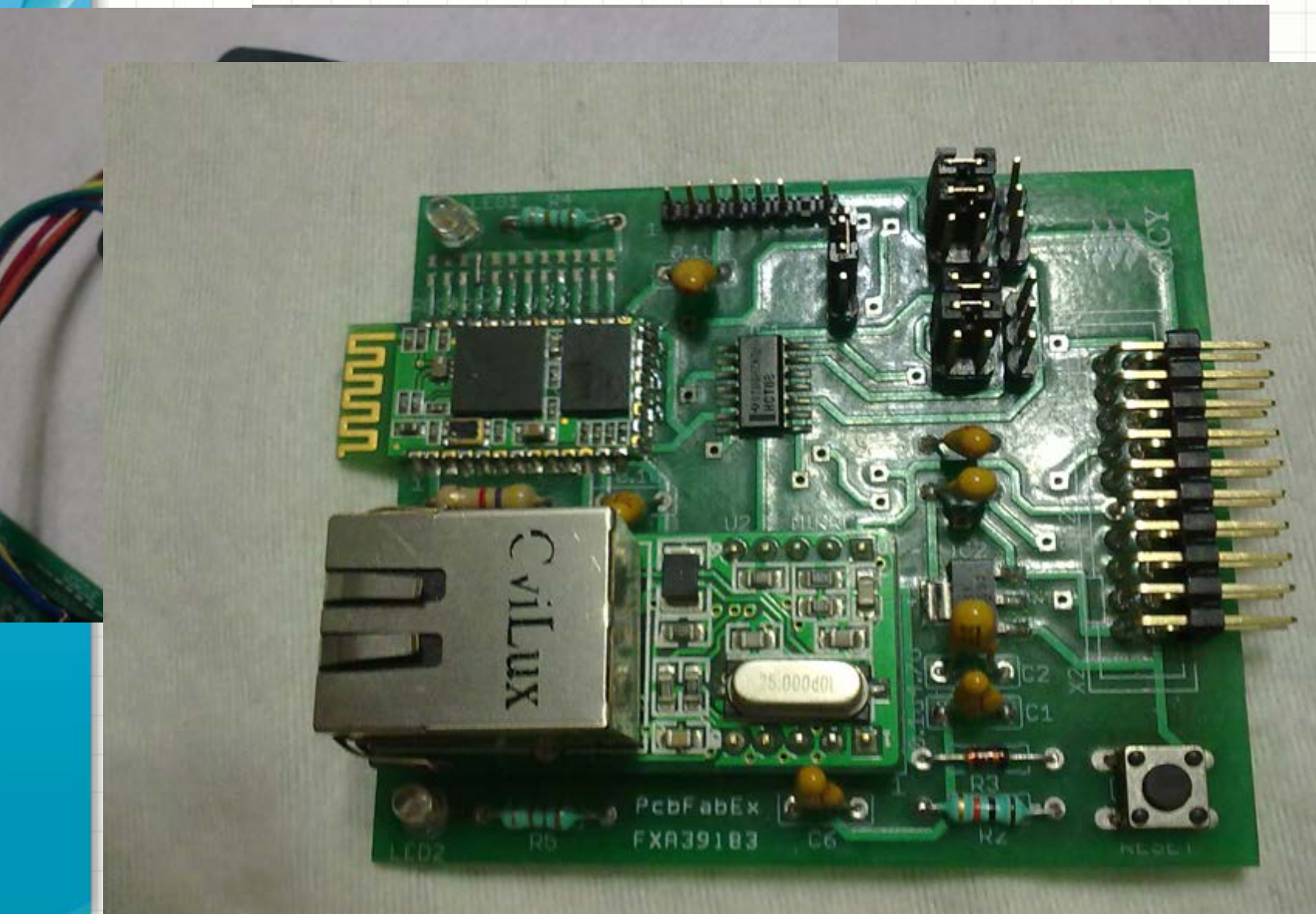
- CUANDO UN RESET OCURRE EL PUNTERO DEL PROGRAMA SE DIRIGE A LA LOCALIDAD DE MEMORIA 0X0000 COMO LO HARÍA DE NO EXISTIR EL BOOTLOADER. AQUÍ, SI NO; ESTÁ PRESIONADO EL ACTIVADOR DEL BOOTLOADER ESTE REDIRECCIONA A LA DIRECCIÓN
- DONDE SE ENCUENTRA UN NUEVO SALTO HACIA EL BUCLE MAIN DEL PROGRAMA, LO MISMO SUSCEDE CON LOS VECTORES DE INTERRUPCIONES ESTOS ENCUENTRAN SU NUEVO VALOR SUMANDO UN OFFSET DE LA DIRECCION ANTERIOR.

FUNCIONAMIENTO DEL BOOTLOADER

USB configuration descriptor pointers: Son punteros hacia posiciones específicas del área USB descriptors. Son utilizados por el firmware en el momento de la enumeración del dispositivo. Incluyen cantidad de interfaces, consumo de energía, y dentro de las interfaces: cantidad de endpoints, código de clase, de subclase, etc.



Personalizar bootloader



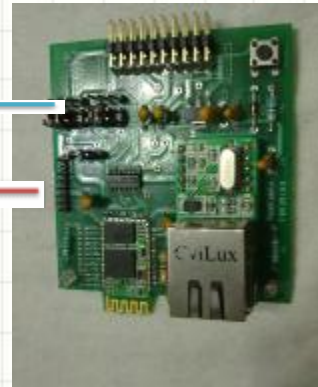
FLASH USB



VDRIVE



TARJETA DE COMUNICACION



ALMACENAMIENTO A MEMORIA FLASH

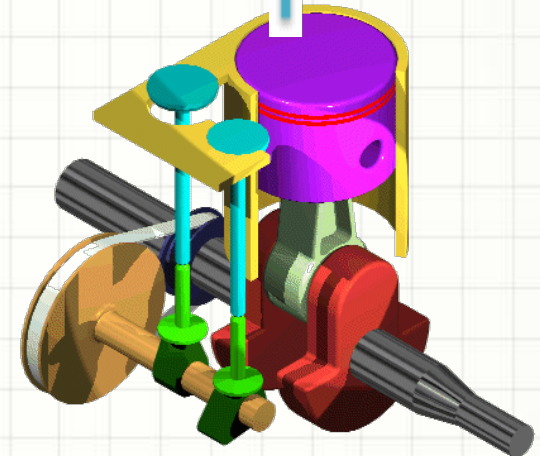


COMUNICACIÓN INALÁMBRICA BLUETOOTH



ALUX V1.1.

CENTRO DE MANDO



PROCESO CONTROLADO