



Aguijón

Notas de aplicación

Nota de aplicación 29:

MCP23S17 16-Bit I/O expander

Descripción:

Mediante el uso de los puertos de expansión como bus SPI controlar los 16 bits de uso general del MCP23S17

Herramientas:

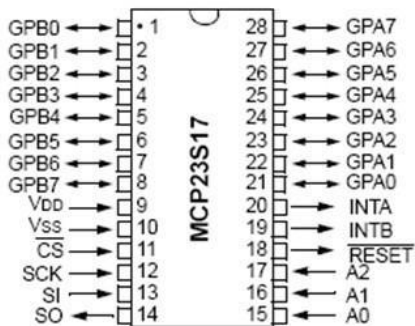
1. Aguijón 3.0, Aguijón 4.0 ó Aguijón 4.1
2. MPLAB X®
3. Aguijón HID bootloader
4. Cable USB 'A' to 'A'
5. Librerías HammerHead.
6. MCP23S17

Pasos:

1. MCP23S17

Puerto bidireccional IO remoto de 16Bits para uso general.

Cuenta con 2 puertos independientes de 8 Bits

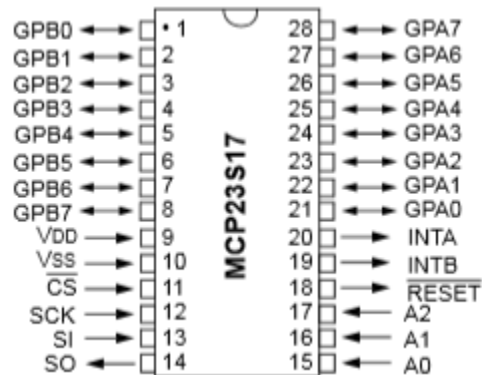


Pin Name	Description
GPA0..7	8-bit General I/O Port A
GPB0..7	8-bit General I/O Port B
INTA	Port A Interrupt Signal
INTB	Port B Interrupt Signal
RESET	Reset Signal
A0,A1,A2	Configurable Address
CS	Chip Select (active Low)
CSK	Synchronous Clock
SI	Slave In
SO	Slave Out
V _{DD}	+5 Volt
V _{SS}	GND



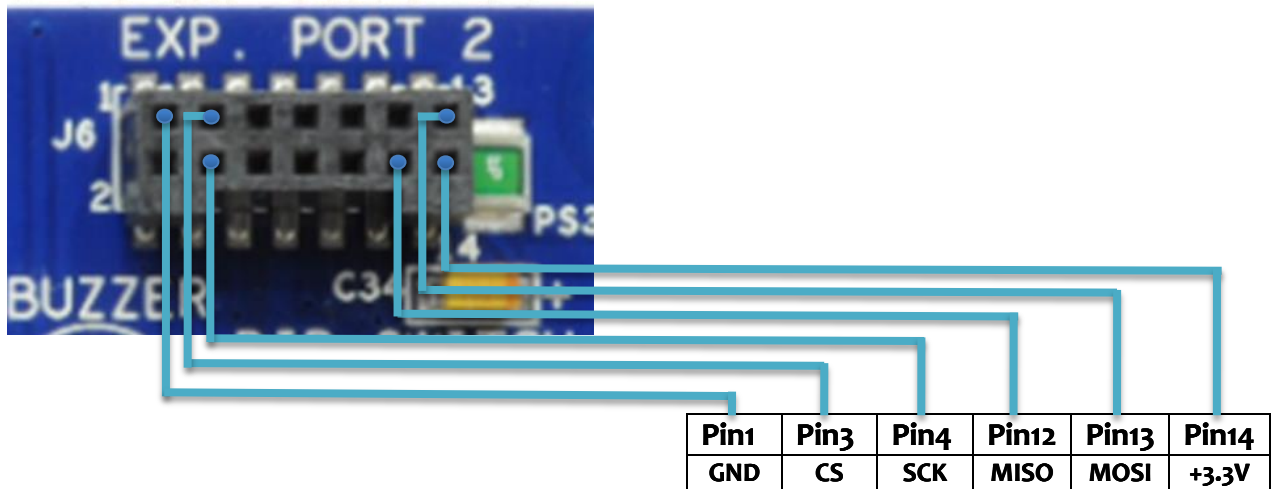
Microchip MCP23S17 SPI 16-bit I/O Expander

2. Diagrama de conexión

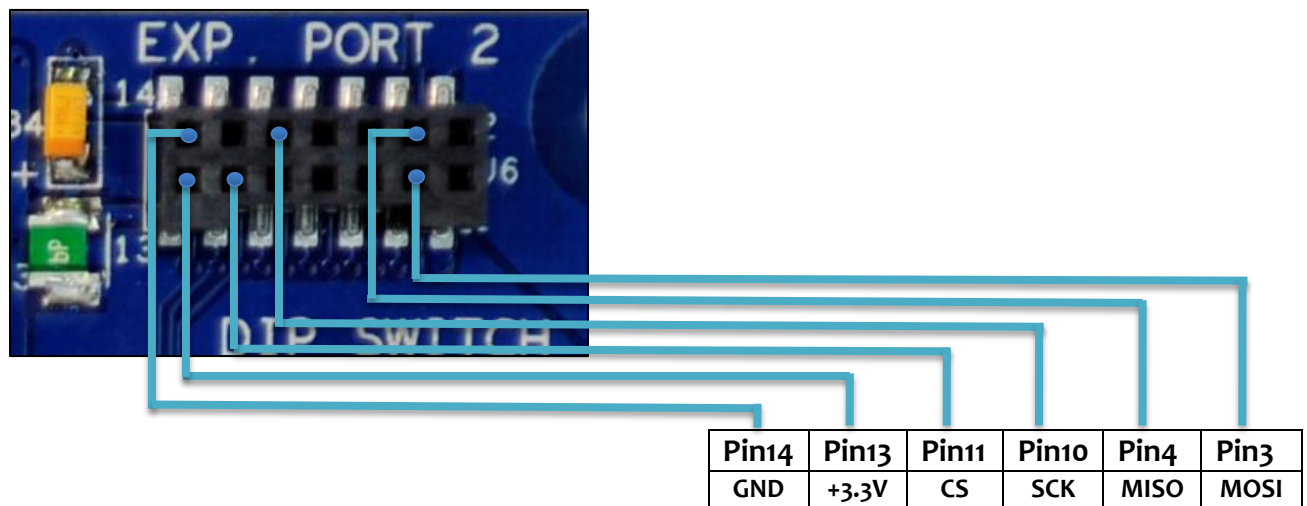


*Los pines **A0**, **A1**, **A2** deben ser conectados a GND, mientras que el pin **RESET** deberá ser conectado a VDD

Puerto de expansión 2, Aguijon 4.0

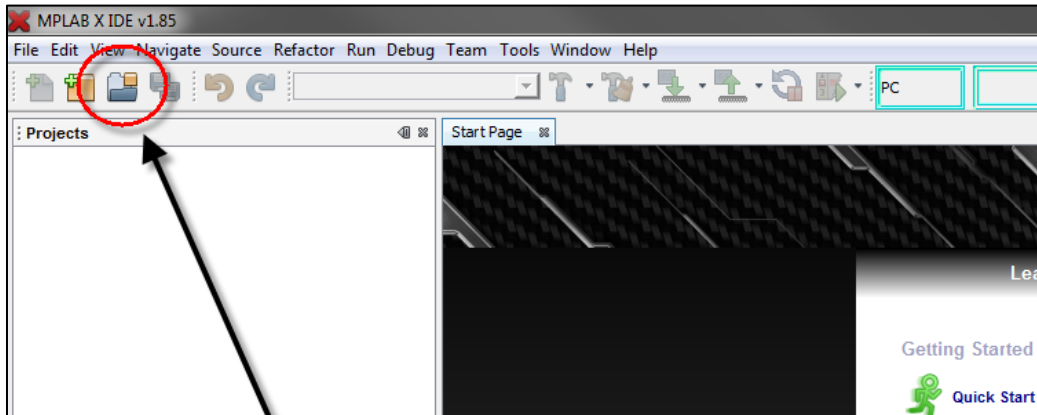


Puerto de expansión 2, Aguijon 4.1



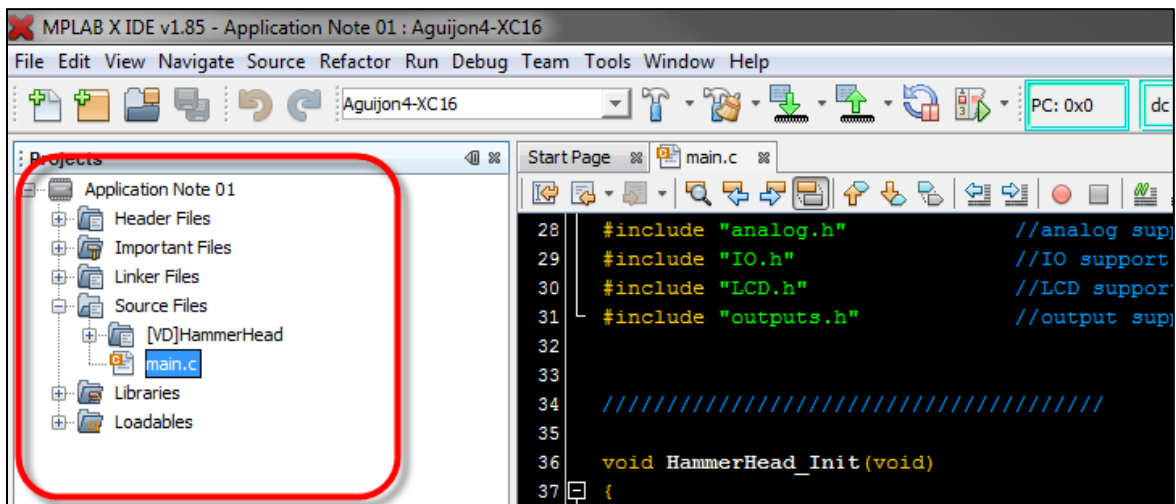
* Ver archivo **Cambios REV4.1**

3. Abrir MPLAB X® y cargar el archivo del proyecto: **Application Note 29.X**



Haz 'clic' aquí y
abre el proyecto

4. Abrir el archivo **main.c**



5. Ir a la línea #50

Utilizaremos la siguiente función:

```
43      /*Inputs*/
44      ADC_Init();
45
46      /*LCD*/
47      LCD_Init(LCD_MODE_1);
48
49      /*MCP23S17*/
50      MCP23S17_Init();
51
52      #if defined(USE_LCD_EXTRA_FEATURES)
53          LCD_BacklightFadeIn();
54      #else
55          LCD_BacklightSet(BLIGHT_LVL_4);
56      #endif
57
```

MCP23S17_Init ();

- Función que inicializa la comunicación SPI y de igual manera ajusta los registros necesarios para la comunicación con el integrado.

Esta función deberá ser llamada antes que cualquier otra relacionada con el MCP23S17.

6. Ir a la línea #74

Utilizaremos la siguiente función:

```
68 HammerHead_Init(); //initialize [VD]HammerHead
69 LCD_IntroAnimation();
70 LCD_PutStr(1,0,"Vinagron Digital",TRUE);
71 LCD_PutStr(2,0,"Application Note 29",FALSE);
72
73 delays(1000);
74 //Port A configured as output
75 MCP23S17_PortsConfig(PORT_A,0b00000000);
76 //Port B configured as input
77 MCP23S17_PortsConfig(PORT_B,0b11111111);
78
79 for(;;){
80
81     for(i=128;i>0;i>>=1)
82     {
```

MCP23S17_PortsConfig(BOOL Port, char direction);

- Función que configura la dirección de los puertos; Donde:
Port = Puerto a configurar (PORT_A ó PORT_B)
direction = Dirección de cada bit del puerto donde: (0=Salida y 1=Entrada)

7. Ir a la línea #84

Utilizaremos la siguiente función:

```
77     MCP23S17_PortsConfig(PORT_B,0b11111111);
78
79     for(;;){
80
81         for(i=128;i>0;i>>=1)
82         {
83             //Read PortB
84             portb=MCP23S17_Read(PORT_B);
85
86             //Create a strings for the LCD//
87             sprintf(lcdMSG,"PortB value= %i ",portb);
88             LCD_PutStr(1,0,lcdMSG,TRUE);
89
90             //Write in Port A & Port B
91             MCP23S17_Write(i,0x00);
```

MCP23S17_Read (BOOL port);

- Función que lee el valor del puerto indicado del integrado de expansión; donde:
port= Puerto a leer (PORT_A ó PORT_B)

8. Ir a la línea #87

Utilizaremos la siguiente función:

```
80
81     for(i=128;i>0;i>>=1)
82     {
83         //Read PortB
84         portb=MCP23S17_Read(PORT_B);
85
86         //Create a strings for the LCD//
87         sprintf(lcdMSG,"PortB value= %i ",portb);
88         LCD_PutStr(1,0,lcdMSG,TRUE);
89
90         //Write in Port A & Port B
91         MCP23S17_Write(i,0x00);
92
93         delayms(100);      //Delay 100 Milliseconds
94     }
```

sprintf (char *, const char *, ...);

- Función que Genera una cadena de caracteres y lo asigna a una variable; Donde:
Char * = Variable a asignar cadena de caracteres (Variable de tipo char)
const char *, = Cadena de caracteres a asignar a la variable.

9. Ir a la línea #88

Utilizaremos la siguiente función:

```
81     for(i=128;i>0;i>=>1)
82     {
83         //Read PortB
84         portb=MCP23S17_Read(PORT_B);
85
86         //Create a strings for the LCD//
87         sprintf(lcdMSG,"PortB value= %i ",portb);
88         LCD_PutStr(1,0,lcdMSG,TRUE);
89
90         //Write in Port A & Port B
91         MCP23S17_Write(i,0x00);
92
93         delayms(100);        //Delay 100 Milliseconds
94     }
95 }
```

LCD_PutStr (int y, int x, char *msg, BOOL clear);

- Función que muestra una cadena de caracteres en la pantalla LCD; donde:
Int y = Coordenada en y (Valor entero del 1 al 2.)
Int X = Coordenada en x (Valor entero del 0 al 20.)
Char *msg = Cadena de caracteres (De 0 a 20 caracteres)
BOOL clear = Determina si se borra la pantalla antes de escribir
(TRUE =Borrar, FALSE =No borrar).

10. Ir a la línea #91

Utilizaremos la siguiente función:

```
84         portb=MCP23S17_Read(PORT_B);
85
86         //Create a strings for the LCD//
87         sprintf(lcdMSG,"PortB value= %i ",portb);
88         LCD_PutStr(1,0,lcdMSG,TRUE);
89
90         //Write in Port A & Port B
91         MCP23S17_Write(i,0x00);
92
93         delayms(100);          //Delay 100 Milliseconds
94     }
95 }
96 return 0;
97 }
98
```

MCP23S17_Write (int data_a, int data_b);

- Función que escribe un valor en los puertos A Y B; donde:
Int data_a= valor que se desea desplegar en el puerto A. (Numero entero del 0 al 255.)
Int data_b= valor que se desea desplegar en el puerto B. (Numero entero del 0 al 255.)

11. Ir a la línea #70.

Utilizaremos la siguiente función:

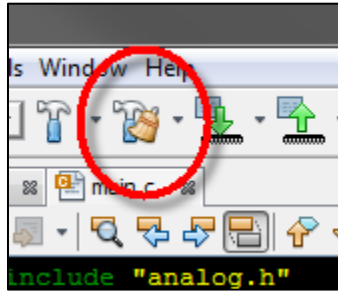
```
84         portb=MCP23S17_Read(PORT_B);
85
86         //Create a strings for the LCD//
87         sprintf(lcdMSG,"PortB value= %i ",portb);
88         LCD_PutStr(1,0,lcdMSG,TRUE);
89
90         //Write in Port A & Port B
91         MCP23S17_Write(i,0x00);
92
93         delayms(100);          //Delay 100 Milliseconds
94     }
95 }
96 return 0;
97 }
98
```

Delayms (ms);

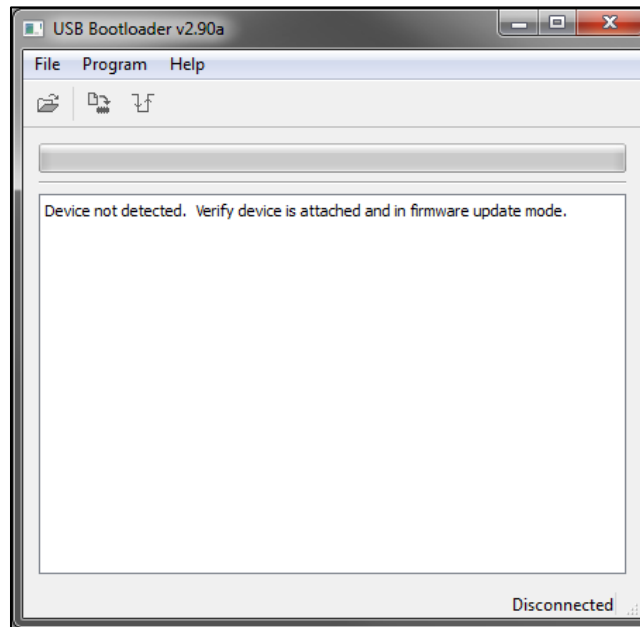
- Función que Pausa el programa por un tiempo determinado (en milisegundos); donde: **ms** = el número de milisegundos que se desea pausar el programa.

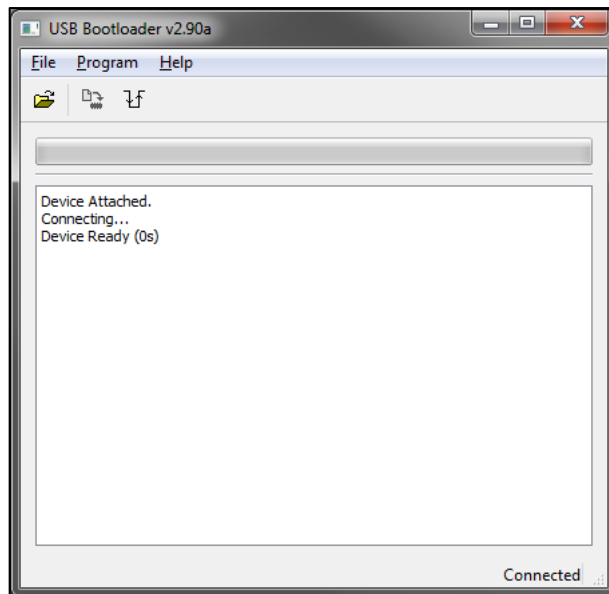
12. Compilar y programar

Al hacer clic en el ícono de compilar, y si no hay errores de compilación, el bootloader será cargado automáticamente.



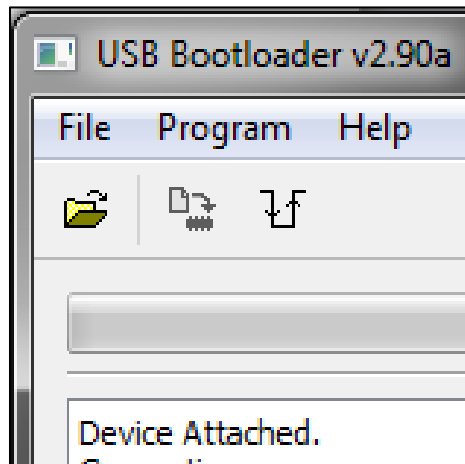
Cuando aparezca la ventana del bootloader, presione el Push-Button número 1 y conecte la fuente de voltaje o encienda el Aguijón y mantenga el PB1 presionado hasta que los LEDs empiecen a parpadear.





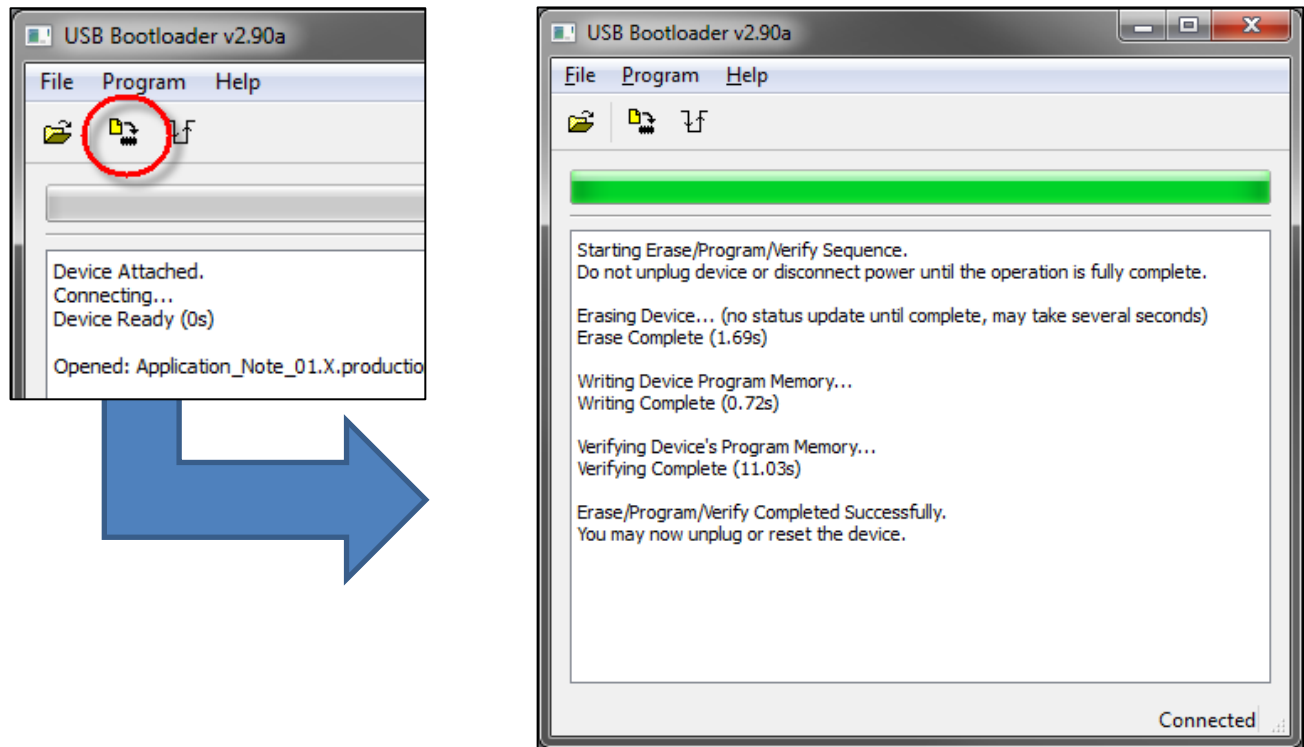
La ventana del Bootloader indicará la conexión establecida con el aguijón:

13. Hacer Clic en Abrir y Cargar el archivo **Application Note 29.X.production.hex**

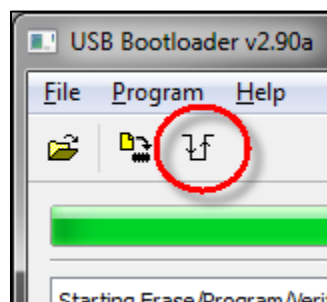


El archivo, depende de la plataforma de hardware.

Una vez cargado el archivo, hacer clic en el ícono de programar.



Hacer clic en el ícono de Reset cuando en la ventana del Bootloader se indique que se terminó de programar con éxito.



Una vez programado podemos verificar el programa corriendo en la tarjeta.

14. Para verificar el funcionamiento del programa verifique que al realizar una buena conexión del integrado (mcp23s17) al puerto de expansión, este genere un corrimiento en sus bits del puerto A, mientras que realiza una lectura en el puerto B, la cual deberá ser desplegada en la pantalla LCD.