**SIN 1 : Maquettage d’une solution en réponse à un cahier des charges**

Document ressource : Les fonctions de la DLL IOWKIT





Sommaire

[1 Starter Kit IOWarrior 24 3](#_Toc286530214)

[1.1 Le composant IOWarrior 3](#_Toc286530215)

[1.2 Carte de prototypage 3](#_Toc286530216)

[1.3 Schéma structurel de la carte de prototypage 4](#_Toc286530217)

[2 Programmation 4](#_Toc286530218)

[3 Présentation de la DLL 4](#_Toc286530219)

[3.1 Les structures de données 4](#_Toc286530220)

[3.1.1 Structure en mode normal 5](#_Toc286530221)

[3.1.2 Structure en mode spécial 5](#_Toc286530222)

[3.2 Les fonctions de la DLL 5](#_Toc286530223)

[3.2.1 Fonction IowKitOpenDevice 6](#_Toc286530224)

[3.2.2 Fonction IowKitCloseDevice 6](#_Toc286530225)

[3.2.3 Fonction IowKitWrite 6](#_Toc286530226)

[3.2.4 Fonction IowKitRead 7](#_Toc286530227)

# Starter Kit IOWarrior 24

## Le composant IOWarrior

Le composant IOWarrior est un contrôleur d’entrées/sorties. Il intègre un certain nombre de fonctions. Ces fonctions dépendent de la version du composant.

Nous allons utiliser le composant IOW 24. Cette version intègre :

* Une interface d’entrées/sorties
* Une liaison I2C
* Une liaison SPI
* Un décodeur infrarouge (code RC5)
* Une gestion de matrice de LED
* Un afficheur LCD compatible avec le HD44780
* 2 registres timers

Toutes ces fonctions se sont pas accessibles simultanément. Le IOW24 dispose de 2 modes de fonctionnement :

* Mode normal : accès aux entrées/sorties
* Mode spécial : accès aux autres fonctions

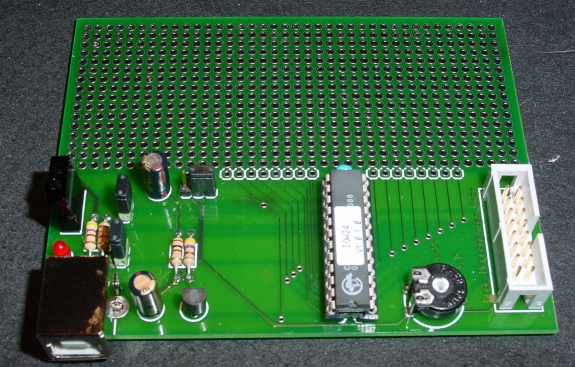
## Carte de prototypage

Le composant est utilisé sur une carte de prototypage.

L’utilisation de la carte peut nécessiter un courant important ou non. Il est dons possible de régler la puissance demandée au port USB par l’intermédiaire d’un cavalier J1 :

* En position High : courant maxi jusqu’à 500mA
* En position Low : courant maxi de 100mA

Pour la suite, on placera J1 sur High.



Cavalier J1

Power High/low

Connexion

I2C et SPI

Capteur

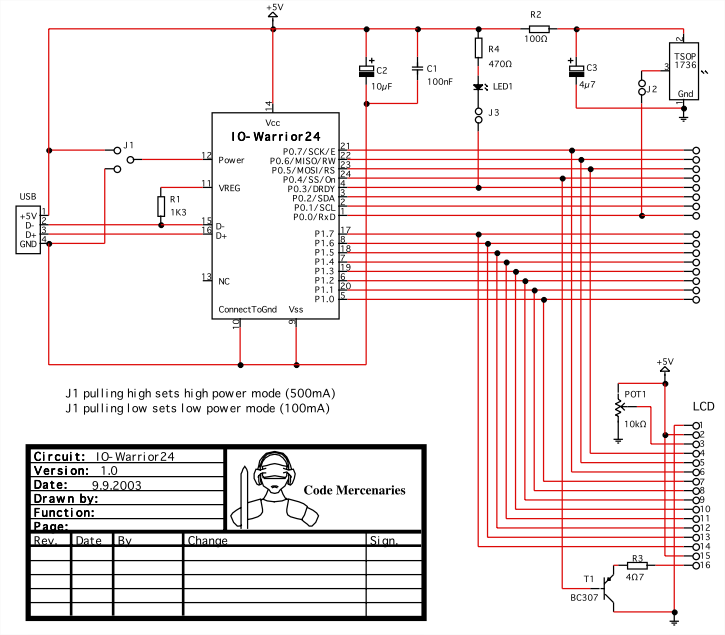
infra-rouge

Connecteur afficheur LCD

Connecteur

USB

## Schéma structurel de la carte de prototypage



# Programmation

Le composant IOWarrior24 livré avec le Starter Kit est programmable de différentes façons :

* Utilisation du logiciel de programmation graphique ProfiLab Expert qui prend en charge une partie des fonctionnalités du composant.
* Utilisation d’un langage de programmation : C, C++, Delphi, Basic.

Le fabriquant du composant fourni une interface de programmation (API) sous forme de DLL utilisable avec les différents langages de programmation

# Présentation de la DLL

La libraire de liens dynamiques (Dynamic Link Library) contient différentes fonctions permettant la commande des différents fonctionnalités du composant IOW24. Pour utiliser la DLL en langage C ou C++, il faut 3 fichiers :

* Iowkit.lib : contient la liste des fonctions
* Iowkit.dll : contient les fonctions
* Iowkit.h : contient le prototype des fonctions, les types personnalisés, les structures...

## Les structures de données

Le composant IOW24 communique avec le PC en utilisant des rapports, des structures de données.

Le composant IOW24 a deux modes de fonctionnement. Chaque mode utilise une interface (pipe ou endpoint) de la liaison USB :

|  |  |
| --- | --- |
| Mode normal | Interface 0 |
| Mode spécial | Interface 1 |

Les rapports sont différents selon le mode de fonctionnement du composant.

### Structure en mode normal

typedef struct \_IOWKIT24\_IO\_REPORT

{

UCHAR ReportID;

union

{

WORD Value;

BYTE Bytes[2];

};

}

IOWKIT24\_IO\_REPORT;

Octet toujours à 0

2 octets contenant les niveaux logiques sur les ports P0 et P1.

Le rapport en mode normal contient 3 octets

### Structure en mode spécial

typedef struct \_IOWKIT\_SPECIAL\_REPORT

{

UCHAR ReportID;

UCHAR Bytes[7];

}

IOWKIT\_SPECIAL\_REPORT;

Octet indiquant l’opération effectuée

Octets de données

Le rapport en mode spécial contient 8 octets

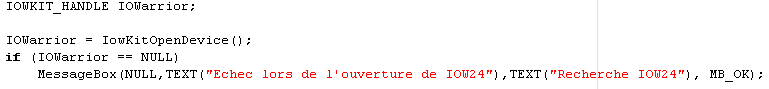
## Les fonctions de la DLL

La DLL contient de nombreuses fonctions. Ne sont détaillées dans ce document que celles qui sont utilisés dans les programmes à réaliser.

|  |  |
| --- | --- |
| **Nom de la fonction** | **Rôle** |
| IOWKIT\_HANDLE IOWKIT\_API IowKitOpenDevice(void) | Ouvrir la communication avec le composant IOW24 détecté sur l’USB. |
| void IOWKIT\_API IowKitCloseDevice(IOWKIT\_HANDLE devHandle) | Fermer la communication avec le composant IOW24. |
| ULONG IOWKIT\_API IowKitWrite(IOWKIT\_HANDLE devHandle, ULONG numPipe, PCHAR buffer, ULONG length); | Ecrire des données au composant IOW24 via l’USB. |
| ULONG IOWKIT\_API IowKitRead(IOWKIT\_HANDLE devHandle, ULONG numPipe, PCHAR buffer, ULONG length); | Lire des données du composant IOW24 via l’USB. |
| ULONG IOWKIT\_API IowKitGetProductId(IOWKIT\_HANDLE devHandle); | Obtenir le numéro d’identification du composant IOW24. |
| ULONG IOWKIT\_API IowKitReadNonBlocking(IOWKIT\_HANDLE devHandle, ULONG numPipe, PCHAR buffer, ULONG length); | Lire des données du composant IOW24 sans bloquer si elles ne sont pas disponibles. |
| BOOL IOWKIT\_API IowKitReadImmediate(IOWKIT\_HANDLE devHandle, PDWORD value); | Lire les valeurs présentes sur les ports d’entrées/sorties du composant IOW24 |

### Fonction IowKitOpenDevice

| **Nom de la fonction** | **Rôle** |
| --- | --- |
| IOWKIT\_HANDLE IOWKIT\_API IowKitOpenDevice(void) | Ouvrir la communication avec le composant IOW24 détecté sur l’USB. |

* Cette fonction renvoie une variable de type IOWKIT\_HANDLE. Ce type est défini dans le fichier d’en-tête ‘*iowkit.h*’. Il correspond à un pointeur universel (sans type défini : void \*). Cette variable permet de distinguer le composant IOW24 d’un autre périphérique.
* IOWKIT\_API est défini comme la commande ‘*\_\_stdcall*’. Elle permet de gérer les paramètres de la pile lors de l’appel de la fonction.
* Si la fonction renvoie la valeur NULL, cela signifie que la communication n’est pas ouverte.
* Attention, il faut déclarer une variable de type ‘*IOWKIT\_HANDLE*’.

Exemple :

### Fonction IowKitCloseDevice

| **Nom de la fonction** | **Rôle** |
| --- | --- |
| void IOWKIT\_API IowKitCloseDevice(IOWKIT\_HANDLE devHandle) | Fermer la communication avec le composant IOW24. |

Elle nécessite le passage du paramètre ‘*devHandle*’ permettant d’identifier le composant IOW24.

Exemple :

### Fonction IowKitWrite

| **Nom de la fonction** | **Rôle** |
| --- | --- |
| ULONG IOWKIT\_API IowKitWrite(IOWKIT\_HANDLE devHandle, ULONG numPipe, PCHAR buffer, ULONG length); | Ecrire des données au composant IOW24 via l’USB. |

Cette fonction renvoie une valeur (de type ULONG : entier long non signé) contenant le nombre d’octets écrits.

Elle nécessite le passage de 4 paramètres :

|  |  |  |
| --- | --- | --- |
| Paramètre | Type | Rôle |
| devHandle | Pointeur universel (void \*) | Permet d’identifier le composant IOW24 |
| numPipe | Entier long non signé | Définit l’interface USB à utiliser pour communiquer |
| Buffer | Pointeur sur octet | Contient les données à transmettre (sous forme de rapport) |
| length | Entier long non signé | Indique le nombre d’octets à transmettre |

Exemple :

### Fonction IowKitRead

| **Nom de la fonction** | **Rôle** |
| --- | --- |
| ULONG IOWKIT\_API IowKitRead(IOWKIT\_HANDLE devHandle, ULONG numPipe, PCHAR buffer, ULONG length); | Lire des données du composant IOW24 via l’USB. |

Cette fonction renvoie une valeur (de type ULONG : entier long non signé) contenant le nombre d’octets lus.

Elle nécessite le passage de 4 paramètres :

|  |  |  |
| --- | --- | --- |
| Paramètre | Type | Rôle |
| devHandle | Pointeur universel (void \*) | Permet d’identifier le composant IOW24 |
| numPipe | Entier long non signé | Définit l’interface USB à utiliser pour communiquer |
| Buffer | Pointeur sur octet | Contient les données lues (sous forme de rapport) |
| length | Entier long non signé | Indique le nombre d’octets à lire |

Exemple :

