soirs, vous pouvez le faire à l'aide d'un IO-Expander qui vous offrira cette fonctionnalité au travers du bus I2C.

Le MCP23016 est un IOExpander de 16 bits, il vous permet donc d'obtenir 16 bits de données utilisables indépendamment en entrée ou en sortie.

Comme la plupart des composants sur bus I2C, le MCP23016 dispose de quelques broches qui vous permettent de spécifier l'adresse du périphérique. Une adresse I2C est toujours sur 7 bits et non pas 8 bits, le bit de poids faible étant utilisé pour spécifier s'il s'agit d'une action de lecture ou d'écriture. La datasheet du MCP23016 précise que l'adresse du composant se compose comme suit : `0b0100xxx`.

Les trois « x » étant paramétrable de façon hardware à l'aide de 3 broches prévues à cet effet A0, A1 et A2). Si A2 = GND, A1 = GND et A0 = 5V, l'adresse de la cible sera donc 0x42.

Utilisation sous uClinux

Avec uClinux vous pourrez développer des applications vous permettant d'utiliser des systèmes d'extension connectée au bus i2c. Toutefois il est de la responsabilité du noyau de gérer le matériel, ce ne sera donc pas à vous de coder le protocole I2C à proprement parler.

Dans la distribution uClinux de votre système de développement, vous trouverez sans doute un driver I2C dédié à votre CPU. Il y a fort à parier que votre noyau n'inclus pas ce driver de base, il faut donc lui adjoindre un driver que nous allons charger depuis un shell afin de l'associer dynamiquement au noyau.

Il faut donc :

- recompiler le driver i2c pour S3C44B0X et uClinux,
- coder une application afin d'utiliser le driver i2c,
- charger le driver i2c sur la cible,
- exécuter l'application.

Pour s'interfacer avec le driver, l'application devra dialoguer avec un fichier d'interface présent sur la distribution uClinux pré-chargé sur la carte : */dev/i2c0*. Pour envoyer des données au module *i2c*, il suffit d'écrire dans le fichier */dev/i2c0*, et pour lire des octets en provenance du module *i2c*, il suffit de lire le fichier */dev/i2c0*.

Driver i2c pour S3C44B0X

Le présent chapitre présente le driver i2c_s3c44.o, la façon de le générer et de le charger sur la cible.

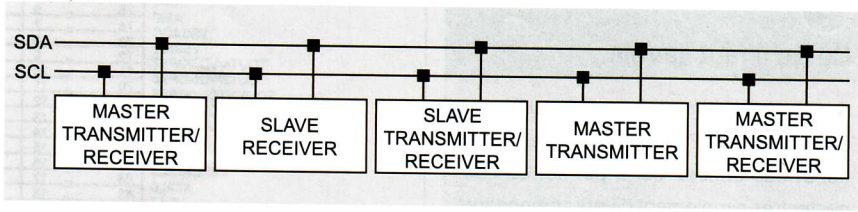

Figure 3. Principe de connexion au bus I2C

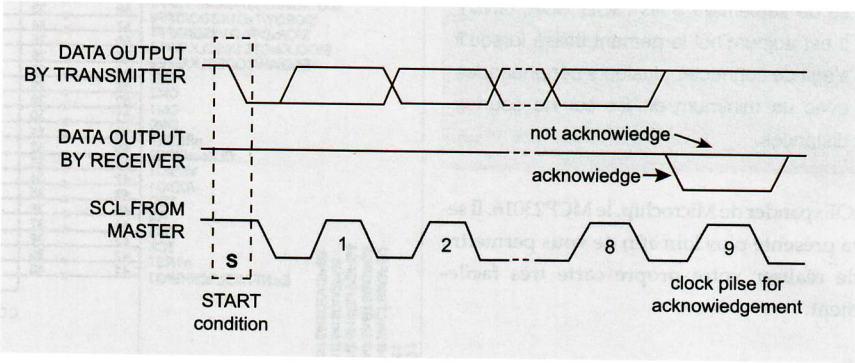

Figure 4. Acquittement du périphérique I2C

Génération du driver

Le driver I2C est un module chargeable dynamiquement sur la cible. Il crée en quelque sorte une extension dynamique du noyau en lui fournissant de nouvelles capacités le temps que le module reste chargé.

Le nom du module I2C est *i2c_s3c44.o*. Il s'agit d'un fichier objet (*.o) et pas d'un binaire exécutable car l'édition de lien est incomplète. Elle sera en fait terminée dynamiquement lors du chargement du module. C'est une des caractéristiques du noyau Linux que d'être capable de lien à la volée du code grâce à une sorte d'éditeur de lien dynamique contenu dans le noyau. Pour générer le driver I2C, allez dans le répertoire *linux-2.4/drivers/i2c/S3C44* de la distribution uClinux de votre carte et compilez le driver à l'aide de la commande `make` : le fichier ainsi généré s'appellera *i2c_s3c44.o*.

Interface /dev/i2c0

Si vous souhaitez communiquer avec des périphériques I2C au travers du driver *i2c_s3c44.o*, il vous faut accéder au fichier */dev/i2c0*.

Les différentes actions possibles sont les suivantes :

- *Open* : il s'agit de la première opération à effectuer sur le fichier. En effet avant de pouvoir effectuer la moindre action sur le driver il convient de demander l'autorisation au noyau. Par exemple si le driver I2C n'est pas préalablement chargé, le noyau vous refusera l'accès au fichier */dev/i2c0*,
- *Close* : c'est le pendant de l'opération *open*. Il permet de signaler au noyau et au driver que l'application ne souhaite plus utiliser le périphérique I2C. Si vous oubliez d'utiliser l'opération *close* à la fin de votre application, le noyau uClinux le fera automatiquement pour vous,
- *Ioctl* : comme nous l'avons vu précédemment, le bus I2C permet d'avoir plusieurs périphériques connectés au bus, il con-

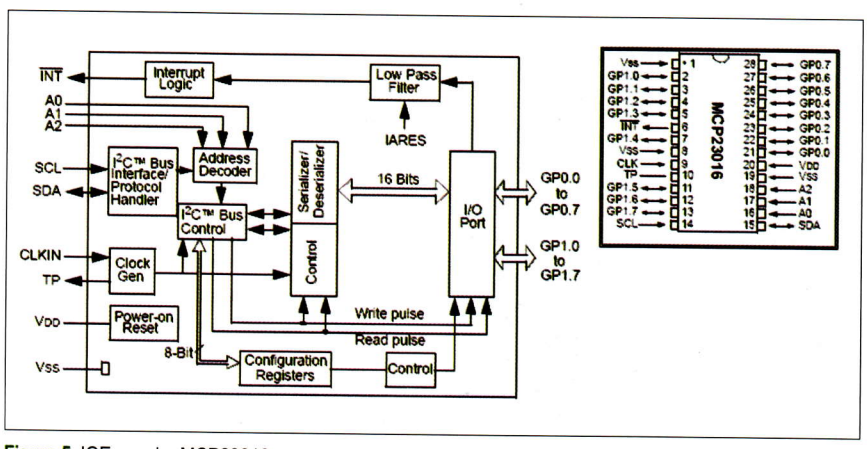

Figure 5. IOExpander MCP23016