REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE SUPERIEURE DES SCIENCES APPLIQUEES
–T L E M C E N–

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
-تلمسان-

École Supérieure En Sciences Appliquées de Tlemcen

Département de la formation du Second Cycle

Filière: Automatique

Polycopie des Travaux Pratiques

Système à micro contrôleur – EasyPIC^{V7}

Élabore Par

Dr MEGNAFI Hicham

Préface

Ce polycopié de travaux pratiques est dédié au module "Système à Microcontrôleur basé sur la carte EasyPIC v7", une ressource spécialement élaborée pour les élèves de 3ème année du cycle ingénieur en spécialité Automatique. Dans un monde où la technologie et l'automatisation jouent un rôle de plus en plus central, il est crucial pour les futurs ingénieurs en automatique d'acquérir une compréhension approfondie des microcontrôleurs et de leur utilisation dans les systèmes automatisés modernes.

Ce module vous offre une opportunité passionnante d'explorer les fondements de l'électronique embarquée, de la programmation de microcontrôleurs et de la création de systèmes automatisés sophistiqués. La carte EasyPIC v7 servira de plateforme pour développer vos compétences pratiques, en vous permettant d'expérimenter avec des composants électroniques, des capteurs et des actionneurs, tout en vous familiarisant avec les concepts clés de l'automatisation.

Ce polycopié de travaux pratiques comprend une série de séances de travaux pratiques, chacune accompagnée d'instructions détaillées pour vous guider dans la réalisation des projets. Vous aurez l'opportunité de programmer des microcontrôleurs, d'interagir avec des capteurs et des actionneurs, et de développer des compétences essentielles dans la résolution de problèmes et la création de systèmes intelligents.

Afin de garantir une expérience d'apprentissage enrichissante et productive, nous aimerions souligner certains prérequis nécessaires pour ce module :

• Connaissances en électronique de base : Une compréhension des composants électroniques tels que les résistances, les condensateurs, les diodes et les transistors est essentielle.

• Programmation : Une connaissance préalable de la programmation et des concepts de base de la logique de programmation vous aidera à saisir plus rapidement les aspects liés à la programmation de microcontrôleurs.

Table des matières

Liste des Figures

Liste des Tables

Introduction générale

I. Introduction aux Systèmes à Microcontrôleur et à EasyPIC v7

I.1 Introduction aux microcontrôleurs

Les microcontrôleurs constituent la basse de la révolution numérique en automatisation industrielle et en technologie embarquée. Ces composants miniaturisés offrent une puissance de calcul et une polyvalence exceptionnelles pour exécuter des tâches spécifiques avec une efficacité maximale. On peut décomposer la structure interne d'un Microcontrôleur en trois parties (voir la figure 1) :

- Les mémoires,
- Le processeur,
- Les interfaces entrées/sorties.

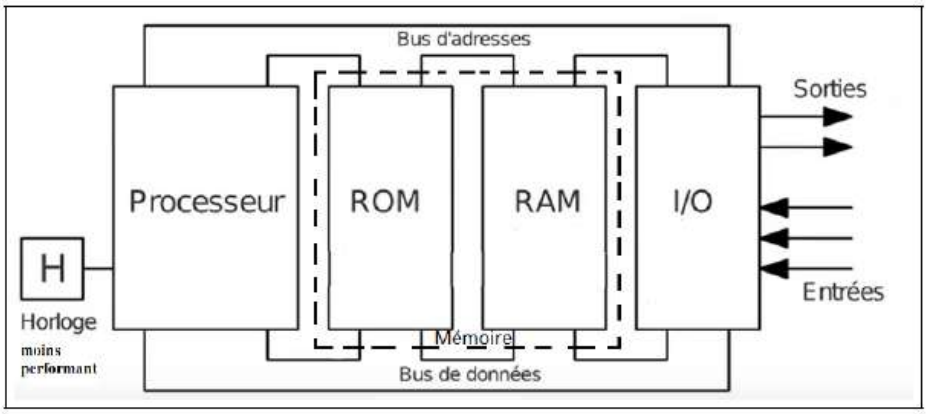


Figure 1. Eléments d'un système à micro contrôleur.

- Les mémoires ont pour fonction de stocker le programme destiné à l'exécution ainsi que les données nécessaires et les résultats obtenus.
- Le processeur représente le cœur du système, étant chargé d'interpréter les instructions du programme en cours d'exécution et d'effectuer les opérations qu'elles contiennent. Au sein du processeur, l'unité arithmétique et logique décode, traduit et exécute les instructions de calcul.
- Les interfaces entrées/sorties ou I/O ont pour rôle de connecter le processeur au monde extérieur dans les deux directions. D'une part, le processeur émet des informations vers l'extérieur (périphérique de sortie), et d'autre part, il en reçoit (périphérique d'entrée).

Pour l'organisation des différentes unités, il existe deux architectures :

- l'architecture Von Neumann
- l'architecture Harvard

I.1.1 Architecture Von Neumann

L'architecture de von Neumann, modèle fondamental de conception d'ordinateurs, utilise une mémoire unique pour stocker à la fois les instructions et les données, avec une séparation implicite entre le stockage et le processeur. Cette approche séquentielle assure une exécution prévisible des instructions, permettant aux ordinateurs d'exécuter divers programmes en chargeant différents ensembles d'instructions en mémoire. Le nom provient de John von Neumann, un mathématicien et physicien qui a formalisé ce concept dans les années 1940. Cette architecture a été cruciale pour le développement des ordinateurs modernes, même si des variantes et des améliorations ont émergé depuis. Le schéma du principe de l'architecture de von Neumann est exposé dans la Figure 2.

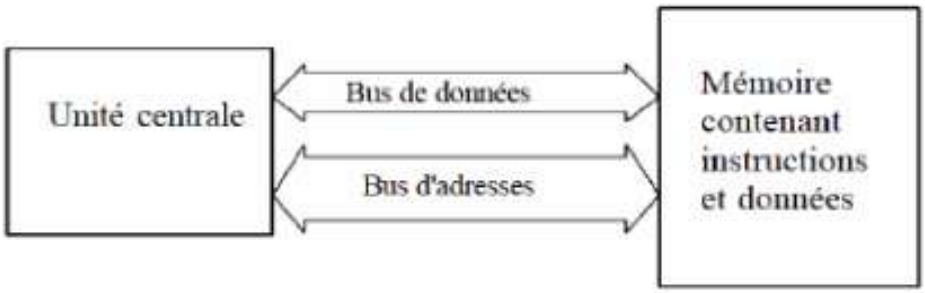


Figure 2. Architecture de Von Neuman.

I.1.2 Architecture Harvard

L'architecture Harvard, modèle informatique, se distingue par sa séparation des mémoires pour instructions et données, permettant un potentiel d'accès parallèle, mais pouvant compliquer la programmation. Utilisée notamment dans les systèmes embarqués pour leur vitesse, son nom provient de la séquence de calculateurs construits à l'Université Harvard, comme le Mark I en 1944. Ce premier ordinateur utilisait une mémoire séparée pour les instructions et les données, marquant l'origine du concept. Cette approche a ensuite évolué, devenant une base pour certaines applications exigeant des performances spécifiques et influençant le développement de nombreuses architectures informatiques modernes. La Figure 3 illustre le principe fondamental de l'architecture de Harvard.

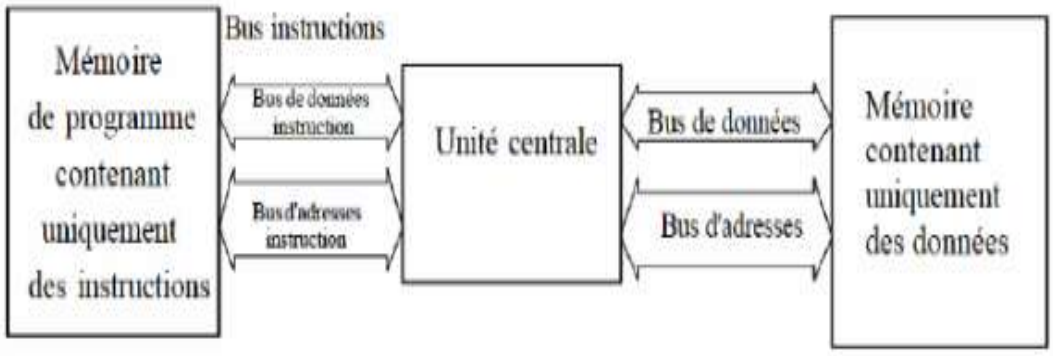


Figure 3.Architecture de Harvard.

I.2 Aperçu des applications courantes des microcontrôleurs dans l'industrie et l'automatique.

Les microcontrôleurs sont omniprésents dans des domaines diversifiés de l'industrie et de l'automatique. Ils constituent la base des systèmes de contrôle, d'acquisition de données et d'interface utilisateur. Dans cette section, nous explorerons comment les microcontrôleurs sont utilisés pour améliorer l'efficacité, la sécurité et la productivité dans ces domaines.

Automatisation Industrielle : Présentation des rôles majeurs des microcontrôleurs dans la surveillance et le contrôle des processus industriels, de la gestion de l'énergie à la gestion des lignes de production.

Contrôle des Machines : Illustration de l'utilisation des microcontrôleurs pour le contrôle précis des machines et des moteurs, optimisant les performances et la régulation.

Systèmes d'Acquisition de Données : Exploration des applications de microcontrôleurs dans l'acquisition, le traitement et la transmission de données provenant de capteurs et d'instruments.

Systèmes Embarqués : Discussion sur la présence des microcontrôleurs dans les dispositifs embarqués tels que les systèmes de surveillance, les appareils médicaux et les objets connecté

I.3 Présentation de la plateforme EasyPIC v7

La plateforme EasyPIC v7 est une solution de développement électronique avancée conçue pour faciliter la création de projets utilisant des microcontrôleurs. Elle est particulièrement adaptée pour la programmation, le prototypage et la mise au point de systèmes embarqués.

Dotée de composants de haute qualité, d'une interface conviviale et de nombreuses fonctionnalités, EasyPIC v7 offre un environnement complet pour les ingénieurs et les développeurs souhaitant réaliser des applications électroniques, allant de la communication série aux interfaces avec des capteurs, en passant par la connectivité sans fil. Cette plateforme est accompagnée d'un logiciel de développement intégré (IDE) et de ressources en ligne pour aider les utilisateurs à concrétiser leurs projets de manière efficace et performante. La figure 4 présente Carte de développement EasyPIC V7 MikroElektronika.

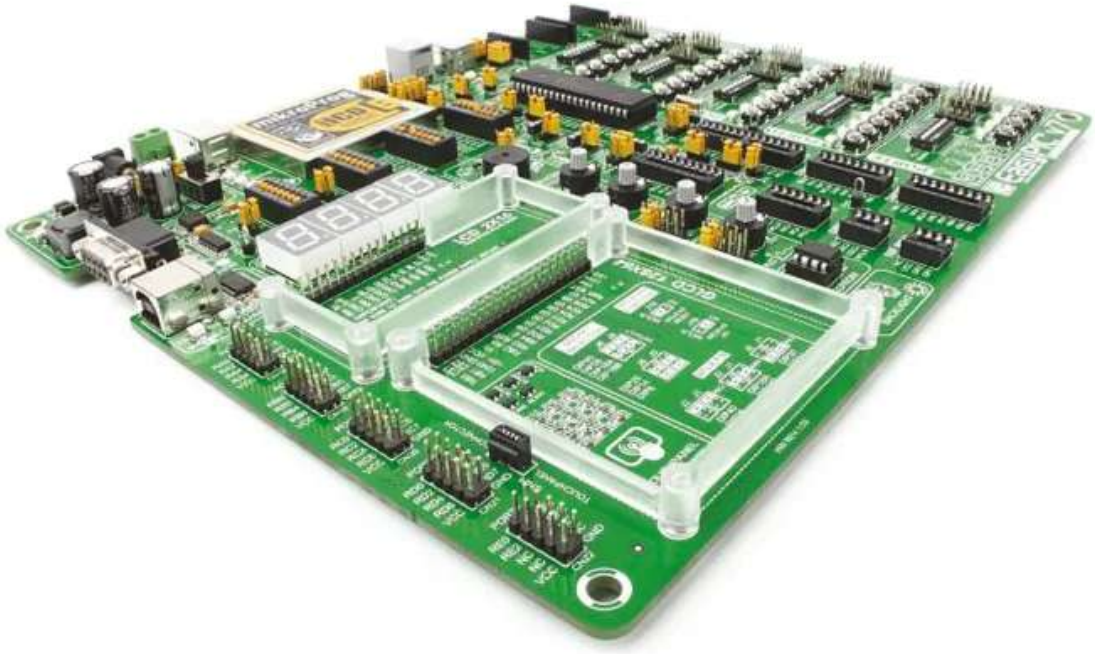


Figure 4. Carte de développement EasyPIC V7 MikroElektronika.

I.3.1 Composants de la plateforme EasyPIC v7

La carte EasyPIC v7 est équipée de plusieurs composants intégrés, chacun jouant un rôle spécifique dans le prototypage et le développement de systèmes embarquée. La figure 5 présente les composants de la carte EasyPIC v7.

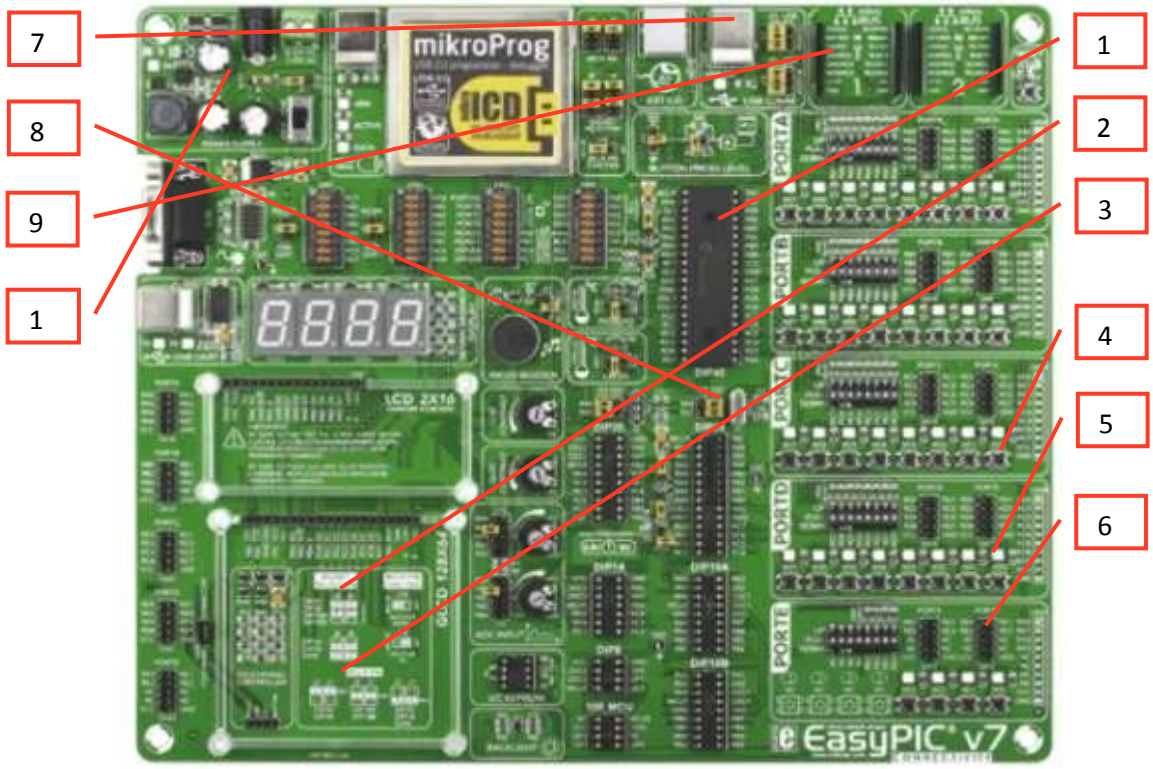


Figure 5. Composants de la carte EasyPIC v7.

Voici une description détaillée des principaux composants de la carte EasyPIC v7 :

1) **Microcontrôleur PIC** : La plateforme EasyPIC v7 est livrée avec un microcontrôleur PIC18F45K22 pré-installé, offrant une puissance de traitement, des capacités de communication et des E/S (Entrée/Sortie) pour interagir avec d'autres composants.
2) **Afficheur Graphique LCD** : Un écran LCD graphique (128x64) rétro éclairé permet d'afficher des informations, des données et des résultats. Il est idéal pour les interfaces utilisateur et la visualisation des données.
3) **Clavier Matriciel** : Un clavier matriciel à 4x4 avec des boutons poussoirs permet l'entrée de données, la navigation dans les menus et l'interaction avec le système.
4) **Boutons Tactiles** : Des boutons tactiles capacitifs offrent une alternative pour l'interaction, facilitant la navigation et les entrées utilisateur.
5) **LEDs Indicatrices** : Plusieurs LEDs (diodes électroluminescentes) sont incluses pour indiquer l'état de différentes opérations et faciliter le débogage visuel.
6) **Interfaces d'Entrée/Sortie (E/S)** : Des connecteurs et des broches E/S configurables permettent la connexion de capteurs, d'actionneurs et d'autres périphériques. Cela offre une flexibilité pour les projets personnalisés.

7) **Interface USB** : La plateforme dispose d'une interface USB permettant la programmation du microcontrôleur, la communication avec un ordinateur et le débogage.
8) **Horloge Temps Réel (RTC)** : Un circuit RTC permet de suivre le temps, ce qui est utile pour les applications nécessitant des horodatages.
9) **Connecteurs d'Extensions** : Des connecteurs sont prévus pour accueillir des modules d'extension, facilitant ainsi la connexion de composants supplémentaires.
10) **Alimentation Intégrée** : Un circuit d'alimentation permet de fournir une tension stable à la plateforme et aux périphériques connectés.

I.3.2 Interface de la carte EasyPIC v7

Les interfaces de la carte EasyPIC v7 sont des connexions configurables qui permettent aux utilisateurs de connecter des périphériques externes, des capteurs, des actionneurs et d'autres composants électroniques pour étendre les fonctionnalités de la plateforme. Voici une description détaillée des principales interfaces de la plateforme EasyPIC v7 :

Broches E/S Configurables : La plateforme EasyPIC v7 dispose de plusieurs broches d'Entrée/Sortie (E/S) configurables qui peuvent être programmées pour différentes fonctions, telles que l'entrée numérique, l'entrée analogique, la sortie numérique, la sortie PWM, etc. Ces broches offrent une grande flexibilité pour interagir avec des dispositifs externes.

Interface de Communication (SPI, I2C, UART) : La plateforme peut offrir des broches E/S dédiées pour les protocoles de communication tels que SPI (Serial Peripheral Interface), I2C (Inter-Integrated Circuit) et UART (Universal Asynchronous Receiver-Transmitter). Ces interfaces permettent de communiquer avec des dispositifs externes, tels que des écrans, des capteurs, des mémoires, etc.

Boutons et Interrupteurs : Bien que principalement utilisés comme des entrées, les boutons et interrupteurs physiques présents sur la plateforme peuvent également être considérés comme des interfaces E/S, permettant à l'utilisateur d'entrer des commandes ou de déclencher des actions.

LEDs Indicatrices : Les LEDs présentes sur la plateforme peuvent être considérées comme des sorties E/S. Elles permettent d'indiquer l'état de différentes fonctionnalités ou de créer des signaux visuels en fonction du programme exécuté.

Connecteurs d'Extensions : La plateforme peut être équipée de connecteurs spécifiques pour accueillir des modules d'extension. Ces connecteurs peuvent inclure des broches E/S supplémentaires pour permettre la connexion directe de composants externes.

II. Présentation de PIC18F45K22

II.1 Description et architecture externe

Le microcontrôleur PIC18F45K22 est un composant électronique hautement performant de la famille PIC18 de Microchip. Son brochage est essentiel pour l'intégration dans un circuit, et comprend des broches spécifiques pour l'alimentation, les entrées/sorties, la programmation, les oscillateurs, et d'autres fonctionnalités. La figure 6 présente les pins de pic PIC18F45K22.

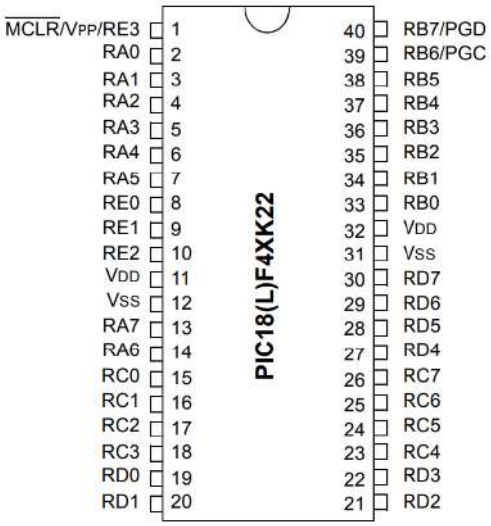


Figure 6. Brochage du circuit intégrer PIC16F84A

Voici une description détaillée du brochage du circuit intégré PIC18F45K22:

Port A (RA0 à RA5) : C'est un port d'E/S de 6 broches qui peut être configuré individuellement comme entrées/sorties numériques ou comme entrées analogiques. Ces broches peuvent également être utilisées pour des fonctions spécifiques, comme l'horloge externe (RA4/TOCKI) ou la sortie de comparaison (RA2/AN2/C2OUT).

Port B (RB0 à RB7) : Le port B est composé de 8 broches d'E/S numériques. Ces broches peuvent également être utilisées pour des fonctions telles que l'entrée du programmateur (PGC) et la sortie du programmateur (PGD), qui sont essentielles pour la programmation et le débogage du microcontrôleur.

Port C (RC0 à RC7) : Le port C est également composé de 8 broches d'E/S numériques, et il est souvent utilisé pour la communication série, comme les lignes d'horloge (SCL) et de données (SDA) pour l'I2C, ou les lignes d'émission (TX) et de réception (RX) pour l'UART.

Port D (RD0 à RD7) : Le port D est un port d'E/S de 8 broches, utilisé principalement pour les fonctions d'entrée/sortie numérique.

Port E (RE0 à RE2) : C'est un port d'E/S de 3 broches numériques, utilisé généralement pour les entrées/sorties numériques.

Broches de réinitialisation (MCLR/VPP) : Cette broche est utilisée pour réinitialiser le microcontrôleur, généralement en la reliant à la masse pour déclencher une réinitialisation. Elle peut également être utilisée pour la programmation en mode série lorsqu'elle est reliée à un programmateur.

Broche d'alimentation (VDD/VSS) : Ces broches sont utilisées pour fournir l'alimentation au microcontrôleur. VDD est la broche d'alimentation positive (typiquement +5V), tandis que VSS est la broche de mise à la terre (0V).

Bouton de réinitialisation externe (MCLR) : Une résistance de tirage peut être utilisée pour connecter une broche à un bouton de réinitialisation externe.

Horloge externe (OSC1/CLKIN et OSC2/CLKOUT) : Ces broches sont utilisées pour connecter un oscillateur externe au microcontrôleur, fournissant ainsi une source d'horloge stable.

Ces broches offrent une grande flexibilité dans la configuration du microcontrôleur PIC18F45K22, permettant de l'adapter à différentes applications et besoins spécifiques. Lors de la conception d'un circuit avec ce microcontrôleur, il est important de prendre en compte la configuration des broches en fonction des fonctionnalités requises.

II.2 Description et Architecture Interne

À l'intérieur du microcontrôleur PIC18F45K22, on trouve une architecture sophistiquée qui lui confère sa puissance et sa flexibilité. Microcontrôleur PIC18F45K22 utilise un bus de données et d'adresse de 8 bits, ce qui définit la taille des transferts de données et la gestion des adresses mémoire. La figure 7 présente Architecture interne de PIC18F45K22.

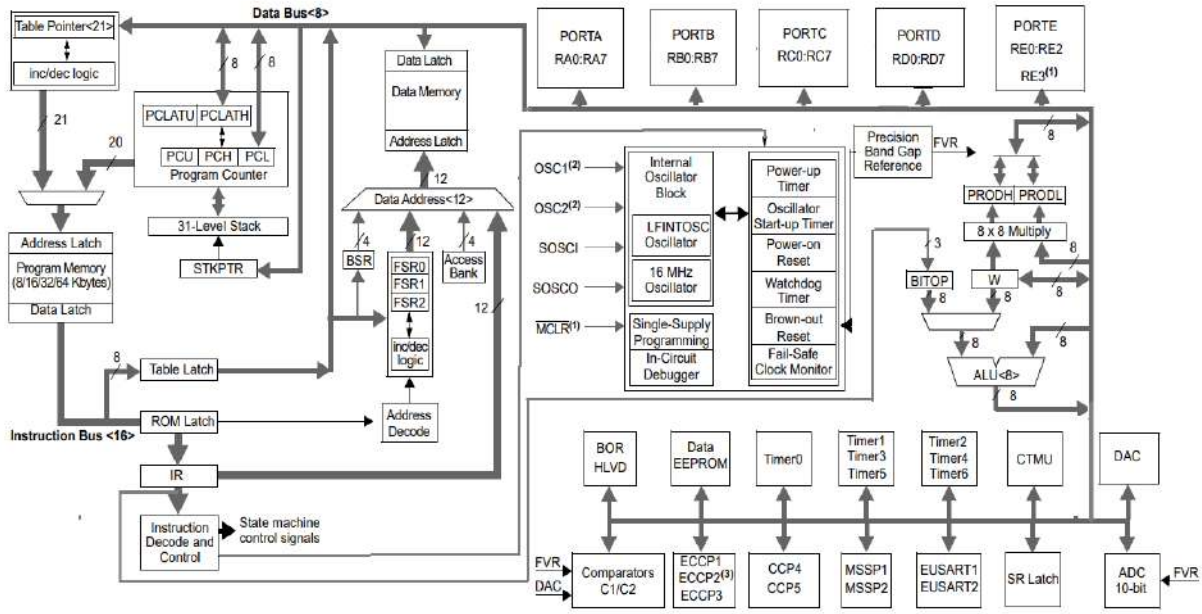


Figure 7. Architecture interne de PIC18F45K22.

Voici une description des éléments clés de son architecture interne :

Unité de Traitement Central (UTC) : Cette unité est le cœur du microcontrôleur, où s'exécutent les instructions du programme. Le PIC18F45K22 utilise une architecture RISC, ce qui lui permet d'exécuter rapidement un ensemble d'instructions bien définies.

Mémoire Flash de programme : Le PIC18F45K22 possède une mémoire Flash de 32 Ko. Cette mémoire est utilisée pour stocker le code exécutable (firmware) de votre application.

Mémoire RAM de données : Le microcontrôleur dispose de 1 024 octets (1 Ko) de mémoire RAM. Cette mémoire est utilisée pour stocker des variables et des données temporaires nécessaires à l'exécution du programme.

Mémoire EEPROM : Le PIC18F45K22 est équipé de 256 octets de mémoire EEPROM (Electrically Erasable Programmable Read-Only Memory). Cette mémoire est non volatile, ce qui signifie que les données y restent même en cas de perte d'alimentation.

Périphériques Intégrés : Le microcontrôleur inclut divers périphériques tels que des temporisateurs, des comparateurs, des convertisseurs analogique-numérique (CAN) et d'autres, permettant une large gamme de fonctionnalités sans avoir besoin de composants externes.

Gestion de l'Alimentation : L'architecture interne gère la consommation d'énergie, permettant au microcontrôleur de fonctionner de manière économe en énergie, ce qui est essentiel pour les applications alimentées par batterie.

Horloge Interne : Le PIC18F45K22 inclut un oscillateur interne qui peut être utilisé comme source d'horloge, simplifiant les conceptions où un oscillateur externe n'est pas nécessaire.

III. Interfaces Numériques, Analogiques de la plateforme EasyPIC v7

III.1 Exploration des interfaces numériques

La plateforme EasyPIC v7 offre des interfaces numériques essentielles qui permettent aux utilisateurs d'interagir avec le microcontrôleur et de contrôler divers dispositifs externes. Les boutons, les interrupteurs et les LEDs sont des éléments fondamentaux de ces interfaces, offrant une interaction simple et efficace.

III.1.1 LEDs

Les LEDs (diodes électroluminescentes) sont des sorties visuelles utilisées sur la plateforme EasyPIC v7 pour indiquer l'état de fonctions, signaler des événements et guider les utilisateurs. Chaque LED peut être connectée individuellement à une sortie de port via un groupe de commutateurs (SW3), permettant de visualiser l'état logique de chaque ligne de port. La Figure 8 présente le schéma de connexion des LEDs avec le PIC18F45K22 et les interrupteurs.

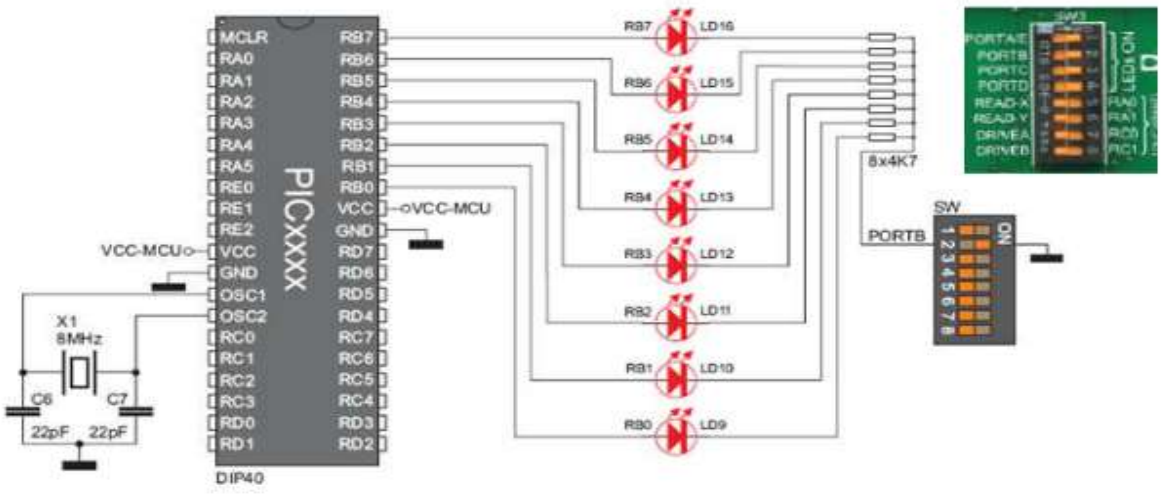


Figure 8. Disposition des connexions des LEDs avec le PIC18F45K22 et les interrupteurs.

III.1.2 Boutons poussoirs relie aux ports

Les boutons poussoirs permettent de modifier l'état logique des entrées numériques du microcontrôleur. Le cavalier J17 est utilisé pour déterminer l'état logique à appliquer à la broche du microcontrôleur lorsque le bouton associé est enfoncé. La Figure 9 illustre les boutons poussoirs reliés au port C.

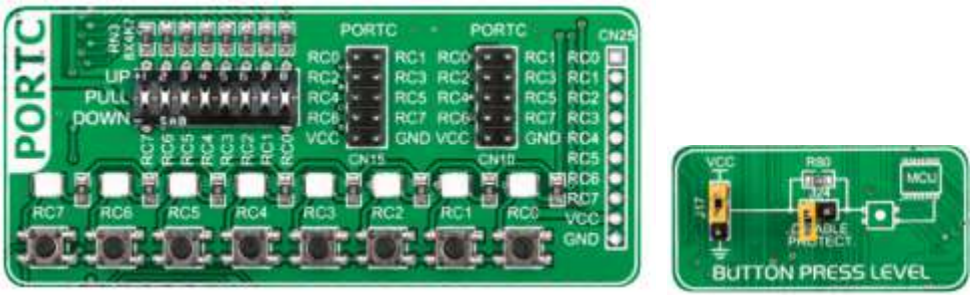


Figure 9. Connexions des boutons poussoirs aux ports.

III.1.3 Interrupteurs

Les interrupteurs associés aux ports permettent d'incorporer des résistances de pull-up ou de pull-down, comme clairement démontré dans la figure 10. Ces résistances sont utilisées pour fixer le niveau logique de chaque entrée lorsque le bouton-poussoir est relâché. L'illustration ci-dessous présente la barrette d'interrupteurs SW7 reliée au port C. Lorsqu'en position médiane, le résistor est déconnecté de la ligne du port. En position haute (UP), un résistor de pull-up est connecté à la ligne du port C correspondante, tandis qu'en position basse (DOWN), un résistor de pull-down est connecté à la ligne du port C correspondante.

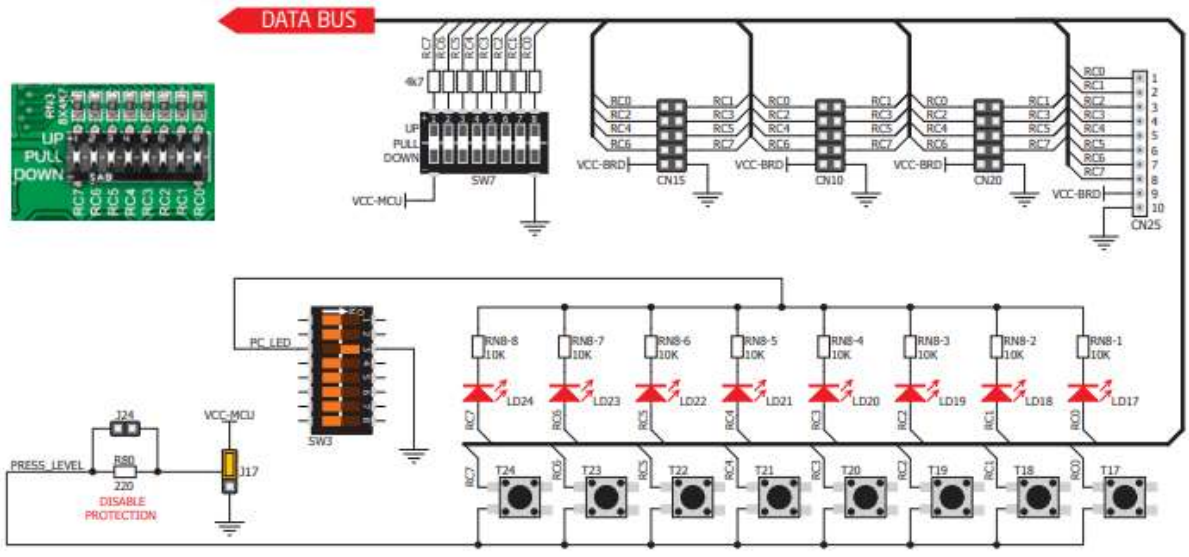


Figure 10. Schéma du groupe d'E/S connecté au port PORTC du microcontrôleur.

III.2 Introduction aux interfaces analogiques

Le microcontrôleur PIC 18F45K22 est équipé d'un convertisseur analogique-numérique (CAN) de 10 bits, qui est connecté aux entrées analogiques via un multiplexeur (permettant la sélection d'une entrée analogique à la fois). Sur la carte EasyPIC V7, il est possible de connecter un potentiomètre P1 ou P2 pour ajuster une tension dans la plage de 0 à 5 volts (Schéma d'entrée de convertisseur analogique-numérique (CAN) est présente dans ma figure 11). Cette tension peut ensuite être reliée à l'une des entrées analogiques, à savoir AN0 à AN4 et AN8 à AN12. Pour effectuer cette connexion, il suffit de placer les cavaliers J15 et J16 selon les indications fournies sur la sérigraphie de la carte.

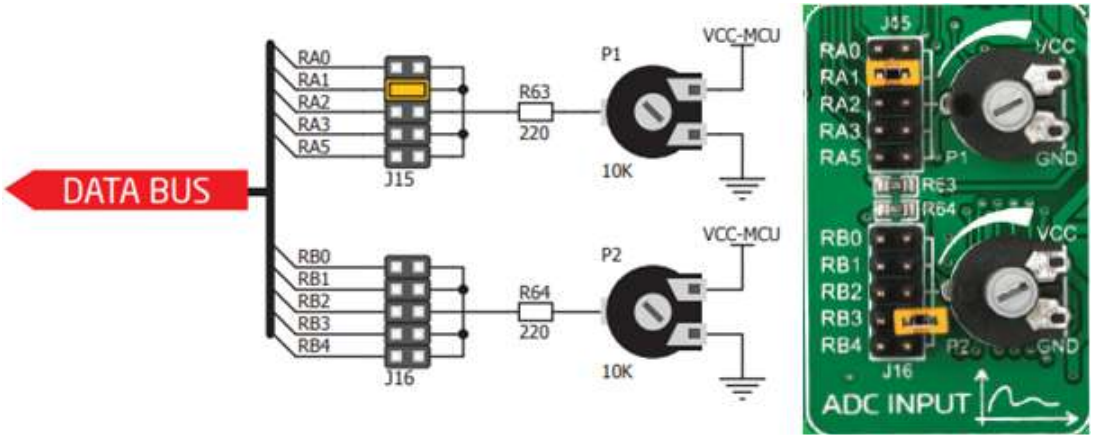


Figure 11. Schéma d'entrée de convertisseur analogique-numérique (CAN)

IV. Communication Série et Protocoles de la plateforme EasyPIC v7

La communication série joue un rôle crucial dans la plateforme EasyPIC V7, offrant des protocoles essentiels tels qu'UART, SPI et I2C, qui permettent aux utilisateurs d'interagir avec une variété de périphériques externes. Ces protocoles sont fondamentaux pour échanger des données de manière efficace, en particulier avec des capteurs et des écrans.

IV.1 Protocole UART

L'UART (récepteur/transmetteur universel asynchrone) est l'une des méthodes les plus courantes pour échanger des données entre le microcontrôleur (MCU) et les composants périphériques. Il s'agit d'un protocole série avec des lignes de transmission et de réception distinctes, et peut être utilisé pour une communication full-duplex. Les deux côtés doivent être initialisés avec le même débit (baud rate), sinon les données ne seront pas reçues correctement.

IV.1.1 UART via RS-232

La communication série RS-232 s'effectue via un connecteur SUB-D à 9 broches et le module UART du microcontrôleur. Pour activer cette communication, il est nécessaire d'établir une connexion entre les lignes RX et TX sur le connecteur SUB-D et les mêmes broches sur le microcontrôleur cible à l'aide de commutateurs DIP. Étant donné que les niveaux de tension de communication RS-232 diffèrent des niveaux logiques du microcontrôleur, il est nécessaire d'utiliser un circuit de transceiver RS-232, tel que le MAX3232. Pour activer la communication RS-232 (figure 12), vous devez placer les cavaliers J3 et J4 en position RS-232, et activer les lignes RX et TX souhaitées via les interrupteurs DIP SW1 et SW2. Par exemple, si vous souhaitez activer la connexion RS-232 sur le module UART1 de la puce PIC18F45K22 par défaut, vous devriez activer les lignes SW1.1 (RC7) et SW2.1 (RC6).

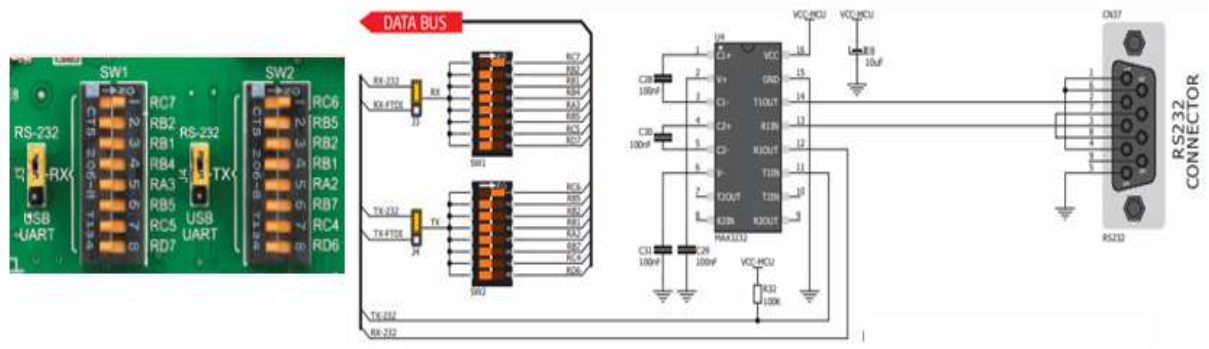


Figure 12. Schéma de connexion RS-232 et paramétrage de l'activation du RS-232 via les interrupteurs SW1 et SW2.

IV.1.2 USB-UART

Les ordinateurs actuels n'ont plus de connecteurs RS-232 ni de contrôleurs UART, car ils sont remplacés par des ports USB et des contrôleurs USB. Cependant, certaines technologies permettent toujours la communication UART via USB. Des contrôleurs comme le FT232RL de FTDI® convertissent les signaux UART en normes USB. Pour utiliser le module USB-UART sur EasyPIC V7, installez d'abord les pilotes FTDI sur votre ordinateur. Cette communication passe par un contrôleur FT232RL, un connecteur USB (CN32) et le module UART du microcontrôleur. Pour établir cette connexion, placez les cavaliers J3 et J4 en position USB-UART et connectez les lignes RX et TX aux broches adéquates du microcontrôleur en utilisant les interrupteurs DIP SW1 et SW2.

Pour activer la communication USB-UART, vous devez placer les cavaliers J3 et J4 en position USB-UART, et activer les lignes RX et TX souhaitées via les interrupteurs DIP SW1 et SW2 (figure 13). Par exemple, si vous souhaitez activer la connexion USB-UART sur le module UART1 de la puce PIC18F45K22 par défaut, vous devriez activer les lignes SW1.1 (RC7) et SW2.1 (RC6).

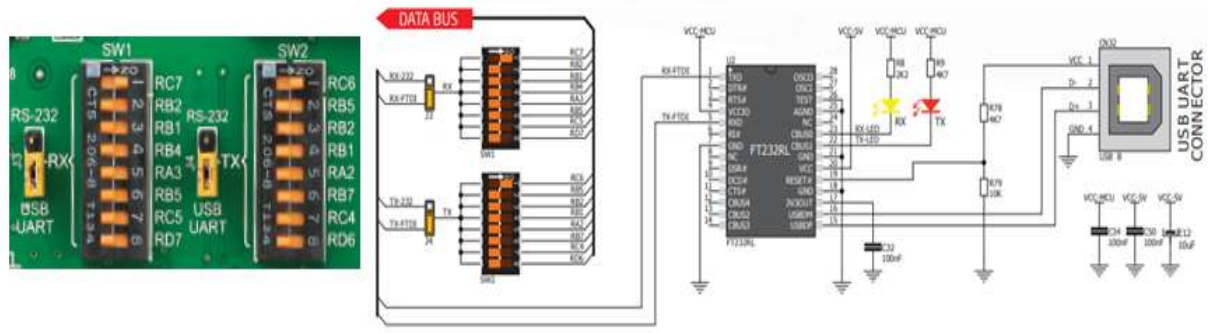


Figure 13. Schéma de connexion USB-UART et paramétrage de l'activation de l'USB-UART via les interrupteurs SW1 et SW2.

IV.2 Protocole I2C

I2C est un bus série multi-maître à simple extrémité utilisé pour connecter des périphériques à faible vitesse à des ordinateurs ou des systèmes embarqués. I2C utilise uniquement deux lignes en drain ouvert, la ligne de données série (SDA) et l'horloge série (SCL), tirées vers le haut avec des résistances. La ligne SCL est pilotée par un maître, tandis que la ligne SDA est utilisée en tant que ligne bidirectionnelle soit par le dispositif maître, soit par le dispositif esclave. Jusqu'à 112 dispositifs esclaves peuvent être connectés au même bus. Chaque dispositif esclave doit avoir une adresse unique.

EasyPIC V7 prend en charge une EEPROM série utilisant une interface de communication I2C et disposant de 1024 octets de mémoire disponible. La carte contient un emplacement pour les EEPROMs sérielles de format DIP8 (figure 14), ce qui vous permet de les échanger facilement avec des circuits intégrés EEPROM de différentes capacités. Le débit de données est de 400 kHz pour une alimentation électrique de 3,3 V ou 5 V.

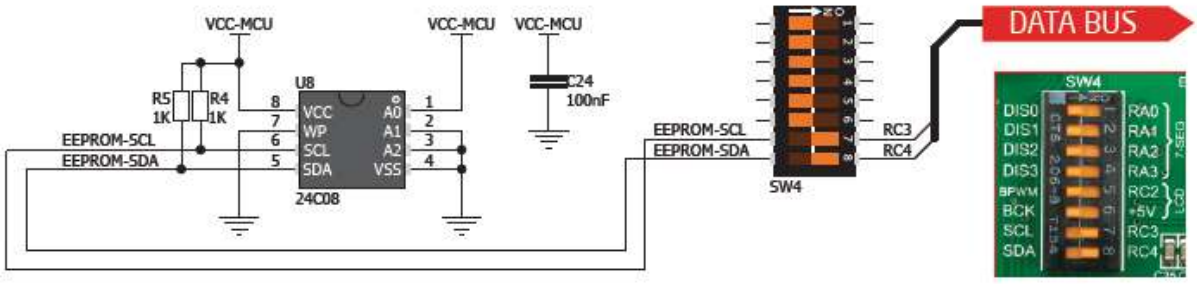


Figure 14. Le schéma du module EEPROM I2C intègre les interrupteurs SW4.7 et SW4.8 pour la connexion des lignes I2C du microcontrôleur à l'EEPROM série

Pour connecter une EEPROM I2C au microcontrôleur, vous devez activer les interrupteurs SW4.7 et SW4.8, comme indiqué sur la figure 16. Les résistances de tirage 1kΩ nécessaires pour la communication I2C sont déjà fournies sur les lignes SDA et SCL une fois que les interrupteurs sont activés. Avant d'utiliser l'EEPROM dans votre application, assurez-vous de déconnecter les autres périphériques, les LED et les résistances de tirage supplémentaires ou de pull-down des lignes d'interface afin de ne pas perturber l'intégrité des signaux/données.

IV.3 Protocole SPI :

Le protocole SPI (Serial Peripheral Interface) est conçu pour une communication série synchrone à haute vitesse. Il est couramment utilisé pour la communication entre microcontrôleurs et périphériques tels que des écrans LCD, des mémoires flash, des capteurs de température, etc. SPI utilise un signal d'horloge maître (SCLK) pour synchroniser la transmission.

V. Applications Avancées et impact industriel avec EasyPIC V7

V.1 Approfondissement des applications avancées

V.1.1 Gestion des interruptions

La gestion des interruptions est essentielle dans de nombreux systèmes embarqués. L'EasyPIC V7 offre une plateforme idéale pour apprendre à gérer les interruptions matérielles. Les microcontrôleurs modernes, tels que ceux intégrés sur cette carte, sont capables de détecter des événements spécifiques (comme des signaux provenant de capteurs) et de déclencher des interruptions pour traiter rapidement ces événements. Cela permet de réduire la latence dans le traitement des tâches importantes. En apprenant à utiliser les interruptions, les développeurs peuvent concevoir des systèmes plus réactifs et efficaces.

V.1.2 Programmation multitâche

La programmation multitâche est cruciale pour des applications avancées qui nécessitent de gérer plusieurs tâches simultanément. L'EasyPIC V7 peut être utilisé pour explorer la programmation multitâche en utilisant des fonctionnalités telles que les temporisateurs, les interruptions et les systèmes d'exploitation en temps réel (RTOS). Les développeurs peuvent apprendre à gérer l'ordonnancement des tâches, à éviter les conflits de ressources, et à assurer une utilisation efficace du temps de traitement. La programmation multitâche est essentielle pour des applications telles que le contrôle industriel, l'automatisation et les systèmes embarqués complexes.

V.1.3 Contrôle de la puissance

La gestion de la puissance est devenue une préoccupation majeure dans le développement d'applications électroniques, notamment pour les systèmes alimentés par batterie ou soucieux de l'efficacité énergétique. L'EasyPIC V7 peut être utilisé pour explorer les techniques de contrôle de la puissance. Cela inclut l'utilisation efficace des modes de faible consommation des microcontrôleurs, la gestion de la consommation des périphériques, l'optimisation de l'alimentation, et la surveillance de la consommation en temps réel. Apprendre à gérer la puissance est essentiel pour prolonger l'autonomie des dispositifs et réduire leur impact environnemental.

V.2 Impact d'EasyPIC V7 dans l'automatisation industrielle et les systèmes de contrôle automatique

L'impact de l'EasyPIC V7 et des microcontrôleurs dans l'automatisation industrielle et les systèmes de contrôle automatique est significatif :

Amélioration de l'efficacité : En utilisant l'EasyPIC V7 et les microcontrôleurs, les entreprises peuvent optimiser leurs processus de production. Les tâches répétitives et les opérations manuelles peuvent être automatisées, ce qui réduit les erreurs humaines, augmente la vitesse des opérations, et permet de réaliser des gains d'efficacité.

Réduction des coûts : L'automatisation permet de réduire les coûts de main-d'œuvre, d'optimiser l'utilisation des ressources et de minimiser les pertes de production dues à des

erreurs. Les microcontrôleurs, notamment lorsqu'ils sont utilisés avec l'EasyPIC v7, offrent une solution économique pour mettre en œuvre cette automatisation.

Qualité et cohérence : Les systèmes de contrôle automatique basés sur l'EasyPIC v7 et les microcontrôleurs garantissent une qualité constante des produits, car les paramètres sont contrôlés de manière précise. Cela est essentiel pour répondre aux normes de qualité de l'industrie et pour maintenir la satisfaction des clients.

Réduction des temps d'arrêt : Grâce à la surveillance en temps réel, les systèmes basés sur l'EasyPIC v7 peuvent détecter les anomalies et les pannes potentielles avant qu'elles ne provoquent des temps d'arrêt coûteux. Cela améliore la disponibilité des machines et la productivité globale.

Flexibilité et évolutivité : Les microcontrôleurs offrent une grande flexibilité pour s'adapter aux besoins changeants de l'industrie. De plus, l'EasyPIC V7 permet de développer rapidement des prototypes et de tester de nouvelles idées, ce qui favorise l'innovation et la capacité à rester compétitif dans un environnement en constante évolution.

Apprentissage et formation : L'utilisation de l'EasyPIC V7 pour explorer l'automatisation et les systèmes de contrôle automatique favorise l'apprentissage et la formation des professionnels de l'industrie. Cela contribue à la création d'une main-d'œuvre qualifiée capable de concevoir, de mettre en œuvre et de maintenir des solutions avancées dans ce domaine.

Travaux Pratiques

Travaux Pratique N°1 : Prise en main de l'environnement de développement pour le PIC18F45K22.

Travaux Pratique N°2 : Manipulation des Entrées/Sorties avec EasyPic V7 : LEDs et Boutons-Poussoirs

Travaux Pratique N°3: Interface Homme-Machine (IHM) avec EsyPIC v7 : Afficheur 7 Segments et LCD128x2

Travaux Pratique N°4 : Implémentation d'un Thermomètre Numérique avec EsyPIC v7 : Acquisition et Affichage des Données

Travaux Pratiques N° 1:

Prise en main de l'environnement de développement pour le PIC18F45K22

I.1 Objectif

- Acquérir une bonne connaissance de l'interface de Proteus ISIS.
- Appliquer le compilateur MikroC PRO pour la programmation d'un PIC.
- Appréhender l'architecture matérielle et logicielle du microcontrôleur PIC18F45K22.
- Maîtriser l'utilisation des boutons-poussoirs en tant qu'entrées numériques.

I.2 Introduction

Il est nécessaire d'employer deux logiciels pour simuler les systèmes embarqués. Le premier de ces logiciels est spécifiquement conçu pour simuler les circuits électroniques, tandis que le second est utilisé pour programmer les PICs. Dans le cadre de ce premier Travaux Pratiques, nous allons utiliser les logiciels suivants : ISIS Proteus et MikroC PRO. Après cela, nous n'utiliserons que la plateforme EasyPIC V7 avec MikroC PRO.

I.2.1 Proteus ISIS

ISIS Proteus est un logiciel développé par la société "Lab Center Electronics" qui propose une simulation électronique complète. Il englobe des simulateurs dédiés à l'analogique, au numérique et à des configurations mixtes pour divers types de circuits électroniques. En utilisant des modules additionnels, ISIS peut également simuler le comportement de microcontrôleurs tels que PIC, Atmel, Intel 8051, ARM, Motorola HC11, et leur interaction avec les composants environnants. C'est cette fonctionnalité particulière d'ISIS qui retient notre attention dans le cadre de ce TP.

I.2.2 MikroC PRO

Le compilateur "mikroC PRO" de "Mikroelektronika" est spécifiquement conçu pour les microcontrôleurs PIC, avec des fonctionnalités intégrées telles qu'un simulateur, un terminal de communication et un gestionnaire de 7 segments. Il prend en charge divers périphériques

industriels courants, tels que les protocoles I2C, 1Wire, SPI, RS485, CAN, les cartes Flash, les signaux PWM, les afficheurs LCD et les afficheurs à 7 segments, offrant une large gamme de bibliothèques et une documentation complète.

I.2.3 Fonction de Bouton-poussoir

Les boutons-poussoirs sont des interrupteurs momentanés simples, employés pour détecter des pressions manuelles. Dans un grand nombre d'applications électroniques, ils assument un rôle fondamental en offrant aux utilisateurs la possibilité d'interagir avec divers dispositifs. Les deux schémas ci-dessous (figure 15) présentent les connexions de boutons-poussoirs avec PIC en utilisant les configurations de pull-up et pull-down.

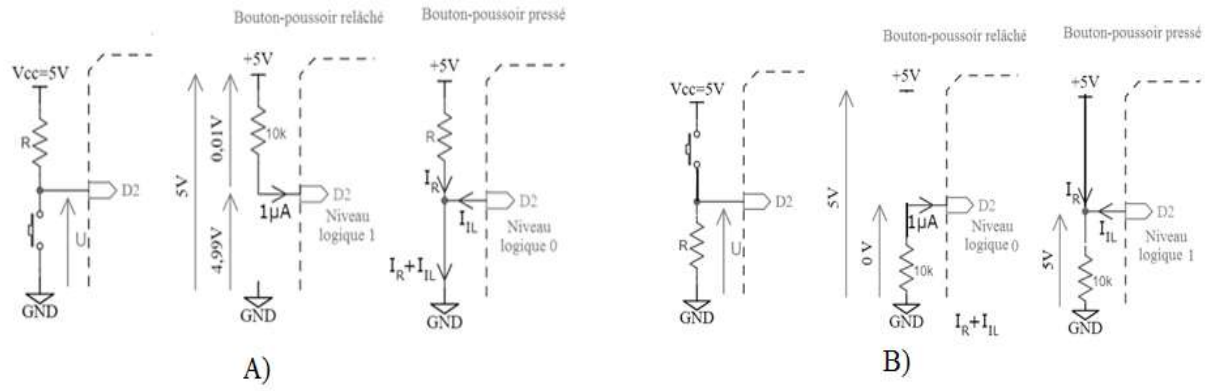


Figure 15. Schéma électrique « bouton-poussoir avec une résistance de tirage »

A) vers le haut (pull-up), B) vers le bas (pull-down).

I.3 Travail à réaliser

Dans cette section, nous entreprendrons la création de quelques manipulations simples en utilisant le microcontrôleur PIC18F45K22. Ces exemples seront extrêmement utiles pour développer une solide compréhension de l'environnement de simulation ISIS ainsi que du compilateur mikroC PRO.

I.3.1 Manipulation 1 : Allumer une LED

Le but de cette première manipulation est d'allumer une LED de couleur rouge connectée à la broche RB0 du port B du PIC18F45K22, suivre les étapes suivantes pour la réalisation:

A. Etape 1 : Configuration de la Simulation

- Démarrez l'environnement de simulation ISIS et créez un nouveau projet.

- Réaliser le circuit de test (voir figure 16) sous ISIS :
  - Ajoutez un composant représentant le microcontrôleur PIC18F45K22 à votre schéma.
  - Intégrez une LED virtuelle dans le schéma et reliez-la à l'une des broches de sortie du microcontrôleur.

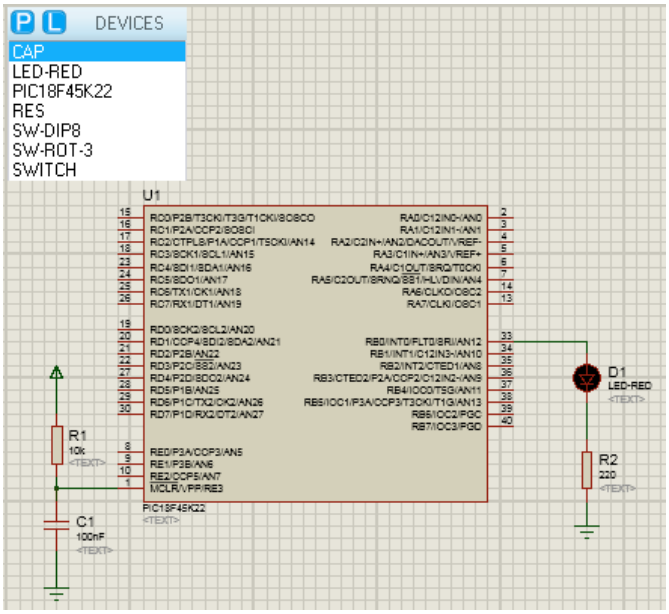


Figure 16. Schéma électronique « commande d'une LED par le PIC18F45K22 ».

B. Etape 2 : Écriture du Code en MikroC PRO

- Ouvrir un nouveau projet sous mikroC PRO.
- Dans le code MikroC PRO, utilisez la fonction appropriée pour configurer la broche de sortie à laquelle la LED est connectée. Par exemple, vous pouvez utiliser la fonction TRISx pour configurer la direction de la broche.
- Dans le compilateur MikroC PRO, écrivez le code ci-dessous :

```
void main() {
trisb=0b00000000;
portb=0;
while (1)
      {portb=0b00000001;}
}
```

C. Etape 3 : Compilation et Génération du Fichier .hex

- Compilez votre code MikroC PRO pour générer le fichier binaire (.hex) correspondant.

D. Etape 4 : Simulation et observation

- Importez le fichier « .hex » dans l'environnement ISIS pour la simulation.
- Exécutez la simulation pour observer le résultat. La LED virtuelle s'allumera en réponse au code que vous avez écrit.

E. Etape 5 : Débogage et Réglages

- Si la LED ne s'allume pas correctement dans la simulation, assurez-vous que le code est correct et que les connexions dans ISIS sont configurées adéquatement.

Dans les applications qui suivent, nous allons refaire les mêmes étapes de 1 à 5, afin de réussir la simulation.

I.3.2 Manipulation 2 : Allumage de 8 lEDs

Pendant cette manipulation, nous simulerons l'allumage de 8 LEDs reliées au port B du microcontrôleur PIC18F45K22. Nous réaliserons la programmation à l'aide du compilateur MikroC PRO, comme indiqué dans le programme ci-dessous. Vous trouverez également le schéma dans ISIS pour cette simulation (figure 17).

```
void main() {
trisb=0b00000000;
while (1) { portb=0b11111111; }
}
```

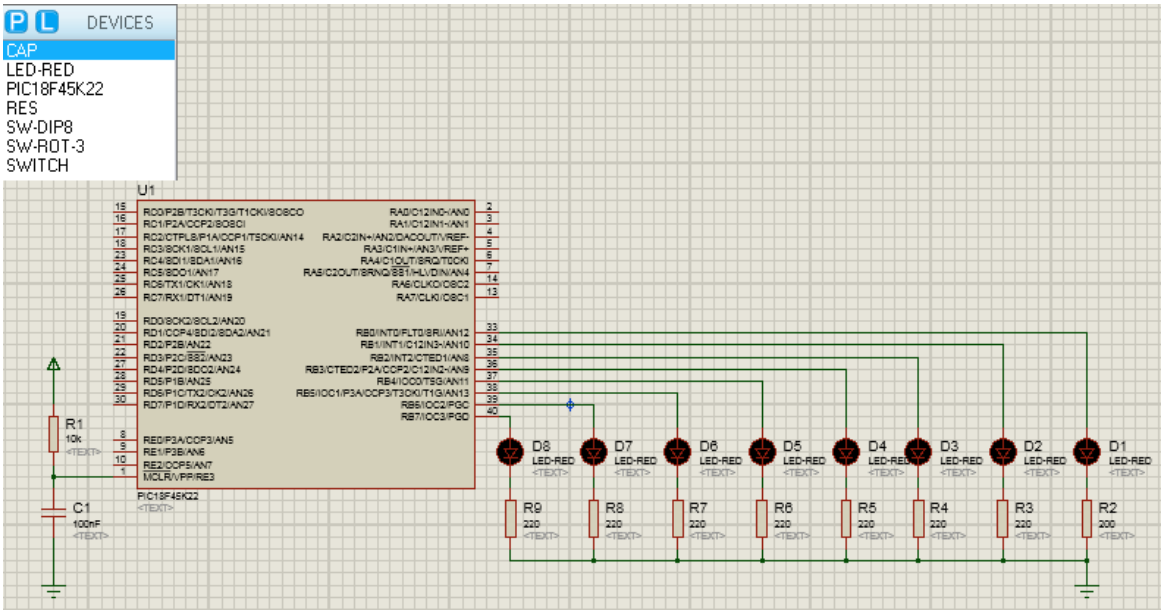


Figure 17. Schéma électronique « commande de 8 LED par le PIC18F45K22 ».

I.3.3 Manipulation 3 : Clignotement de plusieurs LED

L'objectif de cette manipulation est d'organiser un clignotement alterné pour 8 LED de couleur rouge, qui sont connectées au port B du PIC18F45K22. Ces LED sont associées aux broches allant de RB0 à RB7. Chaque LED s'illumine pendant une demi-seconde, suivie d'une extinction pendant la demi-seconde suivante. Il est nécessaire de maintenir en place le montage de test précédent (figure 19). De plus, il vous est demandé d'écrire le programme ci-dessous en utilisant le compilateur MikroC PRO.

```
void main() {
trisb=0b00000000;
while (1)
{ portb=0b00000000;
 delay_ms(500);
 portb=0b11111111;
 delay_ms(500);
 }
}
```

I.3.4 Manipulation 4 : Commandement des 2 LEDs par bouton poussoir avec une résistance de tirage vers le haut (pull-up)

Dans cette manipulation, vous apprendrez à contrôler une LED connectée à la broche RB0 à l'aide d'un bouton-poussoir relié à la broche RD0 (figure 18). Ce bouton-poussoir est relié à une résistance de tirage vers le haut (pull-up). Nous configurerons le bouton-poussoir en tant qu'entrée numérique sur le PIC18F45K22 afin de détecter les pressions et d'allumer/éteindre la LED en conséquence. Le code de la manipulation ainsi que le schéma électronique sont décrits comme suit :

```
void main() {
trisd=0b11111111;
trisb=0b00000000;
while (1)
{ if(portd==0b00000001){portb= 0b00000000;}
 if(portd==0b00000000){portb=0b11111111;}
}
}
```

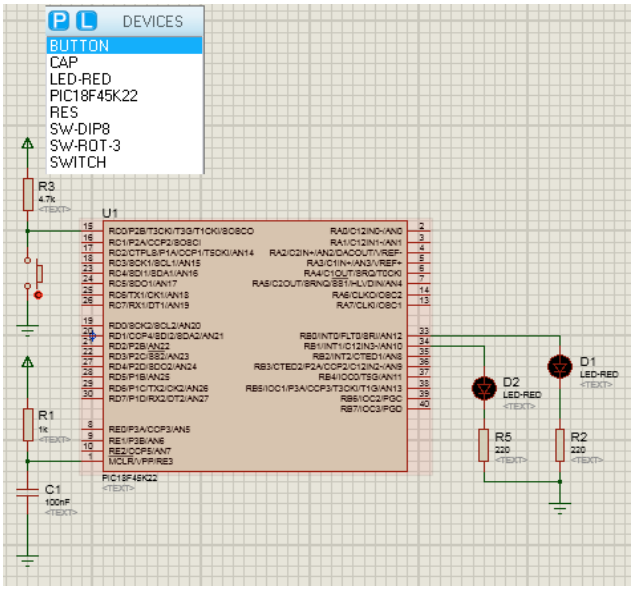


Figure 18. Schéma électronique « commande d'une LED par bouton-poussoir avec une résistance de tirage pull-up ».

I.3.5 Manipulation 5 : Commandement des 2 LEDs par bouton poussoir avec une résistance de tirage vers le bas (pull-down)

À présent, notre objectif est de commander la même LED à l'aide d'un bouton-poussoir connecté à une résistance de tirage vers le bas (pull-down), en conservant le code de manipulation précédent, mais en modifiant le schéma électronique conformément à ce qui est illustré dans la figure 19.

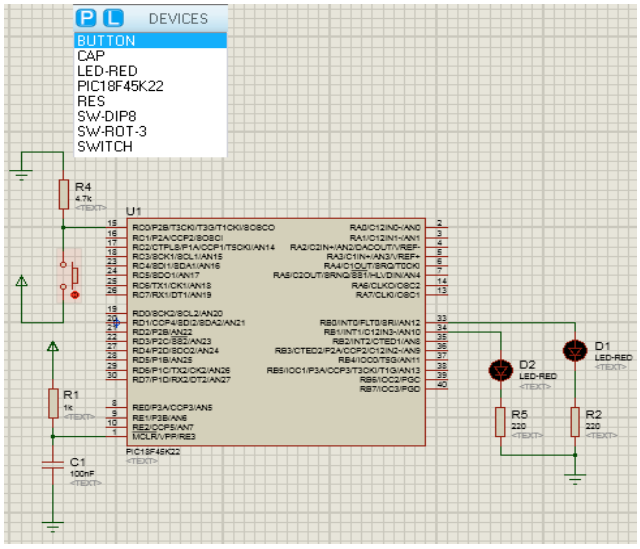


Figure 19. Schéma électronique « commande d'une LED par bouton-poussoir avec une résistance de tirage pull-down ».

- Que pouvez-vous conclure ?

I.3.6 Manipulation 6 : Simulation des I/O des ports B et D de EsayPIC V7

Dans cette manipulation, notre objectif est de simuler le comportement d'un port d'entrée numérique du port B ainsi que d'un port de sortie numérique du port D de la plateforme EasyPIC V7. Pour ce faire, nous avons élaboré un schéma électronique présenté dans la figure 20. Le circuit proposé est composé de plusieurs commutateurs ayant les fonctions suivantes :

- Le commutateur 1 : il permet de connecter ou d'isoler les LED connectées aux ports B de la plateforme EasyPIC V7, étant représenté par le commutateur SW3.
- Le commutateur 2 : il permet de connecter ou d'isoler les LED indiquant les pressions sur les boutons poussoirs reliés aux ports D. Ce commutateur est symbolisé par SW3 sur la plateforme EasyPIC V7.
- Le commutateur 3 : il permet de court-circuiter la résistance de 220 ohms reliée directement aux deuxièmes pattes des boutons poussoirs. Dans la plateforme EasyPIC V7, ce commutateur est représenté par le cavalier J24.
- Le commutateur 4 : il permet de relier les deuxièmes pattes des boutons poussoirs à la masse ou à Vcc. Dans la plateforme, il est désigné par J17.
- Les commutateurs SW1 à SW9 : ils offrent la possibilité de choisir le type de résistance de tirage, pull-up ou pull-down, voire même d'isoler complètement cette résistance. Dans la plateforme, ils sont désignés par SW7.

1. Réalisez le circuit de test (voir la figure 20).
2. Écrivez le programme qui permet de commander les LED connectées aux ports B par les boutons poussoirs, en choisissant le type de résistance de tirage comme pull-up.

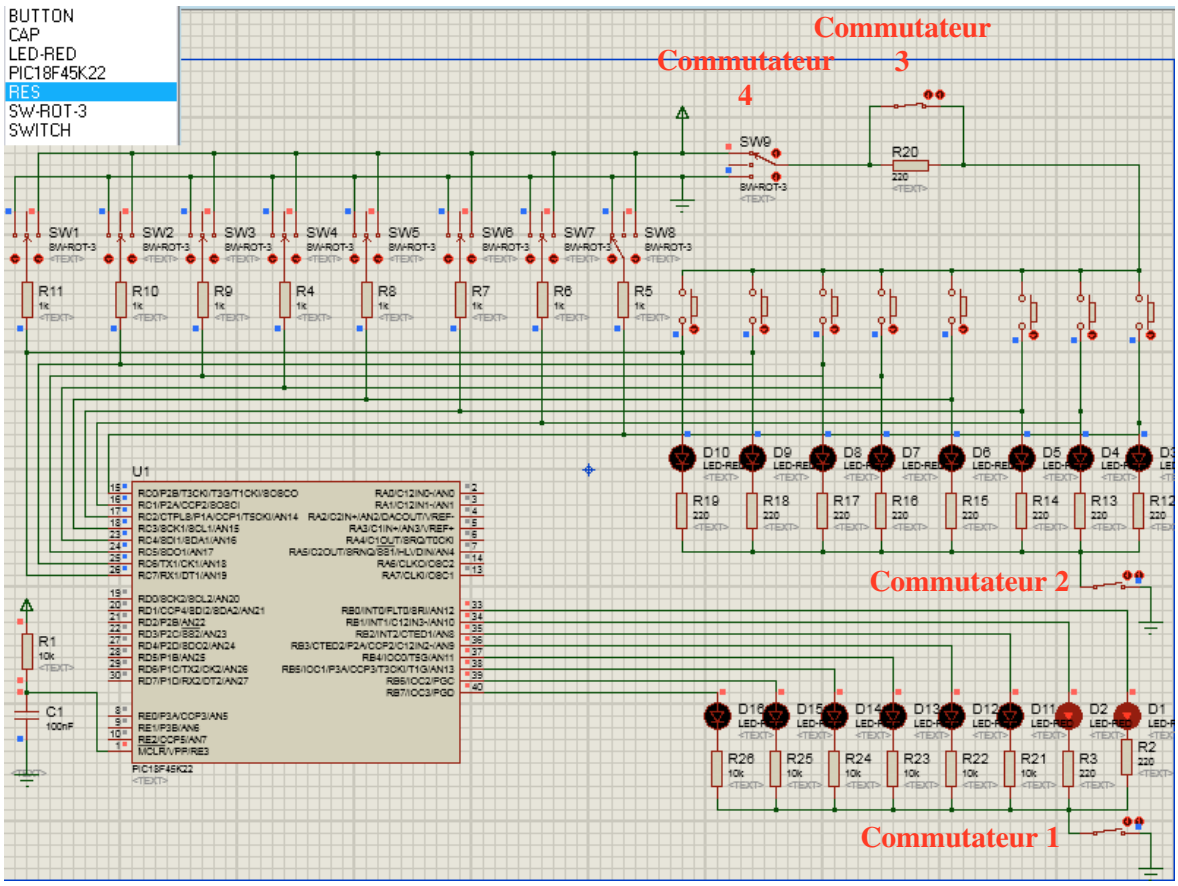


Figure 20. Schéma électronique « Entrés / sorties des ports B et D de EsayPIC V7 ».

Travaux Pratiques N° 2:

Manipulation des Entrées/Sorties avec EasyPIC V7 : LEDs et Boutons-Poussoirs

II.1 Objectif

- Expérimenter le contrôle des entrées/sorties en utilisant la carte EasyPIC V7.
- Acquérir la capacité de programmer les entrées numériques et analogiques ainsi que les sorties pour des applications interactives.
- Comprendre les mécanismes d'interaction entre boutons-poussoirs et LEDs.

II.2 Introduction

II.2.1 Présentation de carte EasyPIC V7

La carte EasyPIC V7 (figure 21) simplifie la création et la programmation de projets basés sur les microcontrôleurs PIC® de Microchip Technology. Elle fournit un environnement convivial pour les ingénieurs et les étudiants en électronique, facilitant la conception, le prototypage et l'exploration d'applications diverses avec les PIC.

Figure 21. Carte EasyPIC V7.

La carte EasyPIC V7 est reconnue pour sa polyvalence en termes de langages de programmation et d'environnements de développement. Parmi les langages pris en charge figure MikroC, offrant une interface conviviale et des bibliothèques préétablies pour simplifier la programmation. En plus de MikroC, la carte est compatible avec MikroPascal et MikroBasic, adaptés respectivement aux utilisateurs de Pascal et de BASIC. Cette gamme de

langages offre une flexibilité optimale pour choisir celui qui correspond le mieux aux besoins de chaque projet.

La carte EasyPIC V7 est équipée d'une gamme de composants périphériques intégrés, tels que des boutons-poussoirs, des LED, des afficheurs, des interfaces de communication (UART, SPI, I2C), des convertisseurs analogique-numérique (CAN), et bien plus encore. Ces fonctionnalités polyvalentes offrent aux utilisateurs un terrain d'expérimentation solide pour tester leurs idées et développer des projets électroniques fonctionnels.

II.2.2 Mise en marche de la carte EasyPIC V7

La mise en marche de la carte EasyPIC V7 implique plusieurs étapes pour s'assurer que la carte est correctement alimentée et prête à être utilisée. Voici comment procéder :

1. **Préparation :** Avant de commencer, assurez-vous d'avoir tous les composants nécessaires, y compris la carte EasyPIC V7, un microcontrôleur compatible et un câble USB.
2. **Connexion USB :** Branchez un câble USB entre la carte EasyPIC V7 et un port USB de votre ordinateur.
3. **Sélection d'alimentation (**cavaliers**) :** Repérez les cavaliers d'alimentation sur la carte, généralement proches de la section d'alimentation. Assurez-vous que les cavaliers sont correctement configurés pour l'alimentation souhaitée. Pour une alimentation via USB depuis l'ordinateur, configurez les cavaliers ainsi : ***Placez le cavalier J6 en position USB et configurez le cavalier J5 en mode 5V***.
4. **Interrupteur d'alimentation :** Localisez l'interrupteur d'alimentation sur la carte EasyPIC V7. Met l'interrupteur sur la position "ON" pour allumer la carte.
5. **Indicateurs LED :** Les LED d'alimentation et éventuellement d'autres indicateurs devraient s'allumer pour indiquer que la carte est alimentée.
6. **Préparation du logiciel :** Si vous prévoyez de programmer la carte, assurez-vous d'avoir installé le logiciel MikroC PRO sur votre ordinateur.
7. **Configuration du projet :** Ouvrez le logiciel de développement, créez un nouveau projet et configurez-le pour le microcontrôleur PIC18F45K22.

8. **Écriture et téléchargement du code :** Écrivez un code de test simple, par exemple pour faire clignoter une LED. Compilez le code et téléchargez-le sur la carte via l'interface de programmation.
9. **Observation des résultats :** Si vous avez programmé une LED pour clignoter, observez si elle clignote conformément au code.

II.3 Travail à réaliser

Dans cette série de manipulations, vous explorerez des concepts fondamentaux de contrôle d'entrées/sorties sur la carte EasyPIC V7. Vous découvrirez comment créer des effets visuels interactifs à l'aide de divers éléments tels que les LEDs, les boutons-poussoirs, les potentiomètres et les délais. Chaque manipulation offre une opportunité unique de développer vos compétences en programmation embarquée et d'expérimenter des interactions entre les composants électroniques intégrés

II.3.1 Manipulation 1 : Décalage de l'Allumage des 8 LEDs de Gauche à Droite

Dans cette manipulation en deux parties, vous allez travailler sur le contrôle de l'allumage des 8 LEDs de gauche à droite avec un effet de décalage. Chaque LED s'allumera successivement en suivant un mouvement de gauche à droite, créant ainsi un effet visuel de défilement. La vitesse de décalage entre chaque LED sera de 100 millisecondes.

A. Partie 1 : Allumage Initial des 8 LEDs

a) Étape 1 : Configuration matérielle

- Identifiez les broches associées aux 8 LEDs sur la carte EasyPIC V7 (figure 22).
- Configurez les broches associées aux LEDs en tant que sorties numériques.

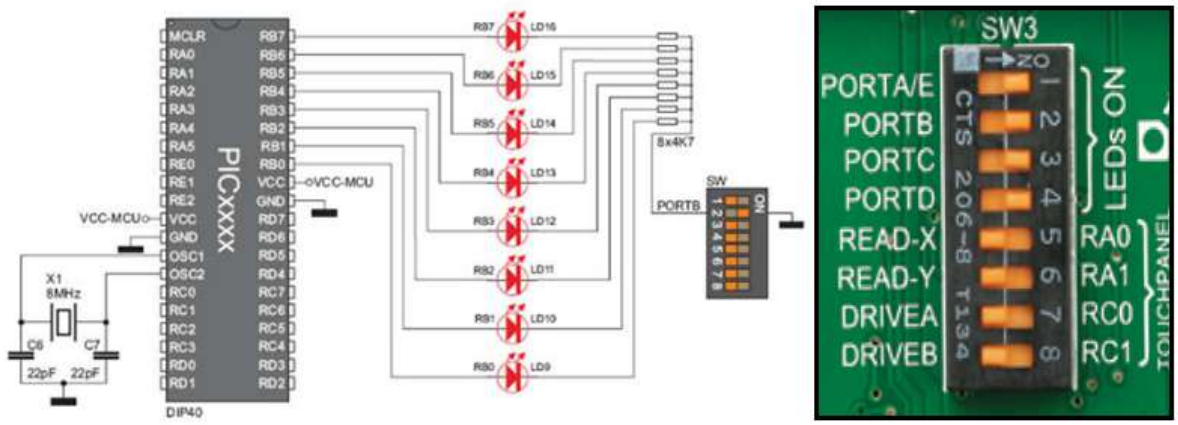


Figure 22. Description de connexions des 8 LEDs avec le PIC.

b) Étape 2 : Programmation en Mikro C pour Allumage des LEDs

- Dans le programme, utilisez une boucle for pour allumer simultanément les 8 LEDs.
- Utilisez la fonction **GPIO_Set** pour allumer toutes les LEDs en même temps.
- Après les avoir toutes allumées, introduisez un délai d'une seconde (1000 millisecondes) avant de passer à la deuxième partie de la manipulation.

B. Partie 2 : Décalage des LEDs

a) Étape 3 : Programmation en Mikro C pour Décalage des LEDs

- Reprenez le code de la Partie 1 et ajustez-le pour créer l'effet de décalage des LEDs de gauche à droite.
- Utilisez une boucle for pour allumer chaque LED successivement de gauche à droite.
- Utilisez la fonction **GPIO_Set** pour allumer une LED et **GPIO_Clear** pour l'éteindre.
- Introduisez un délai de 100 millisecondes (utilisez **Delay_ms**(100)) entre l'allumage et l'extinction de chaque LED.
- Répétez le décalage jusqu'à ce que toutes les LEDs aient été allumées et éteintes.

II.3.2 Manipulation 2 : Utilisation des Boutons-Poussoirs pour Contrôler les LEDs

Dans cette manipulation, vous allez explorer l'utilisation des boutons-poussoirs en tant qu'entrées numériques sur la carte EasyPIC V7. L'objectif est de détecter l'appui des boutons-poussoirs et de contrôler l'allumage des LEDs intégrées sur la carte. Vous allez créer une interaction où la première pression d'un bouton allume les 8 LEDs, tandis que la seconde pression éteint les LEDs. Dans un premier temps, vous allez configurer les boutons-poussoirs sans utiliser de résistances pull-up ou pull-down. Ensuite, vous allez introduire l'utilisation de la résistance pull-up, puis de la résistance pull-down.

A. Étape 1 : Configuration matérielle

- Repérez les boutons-poussoirs intégrés sur la carte EasyPIC V7.
- Identifiez les broches associées à chaque bouton-poussoir en vous référant à la Figure 23. Cette figure présente le schéma électrique des boutons-poussoirs en relation avec les LEDs, les résistances pull-up et les résistances pull-down.

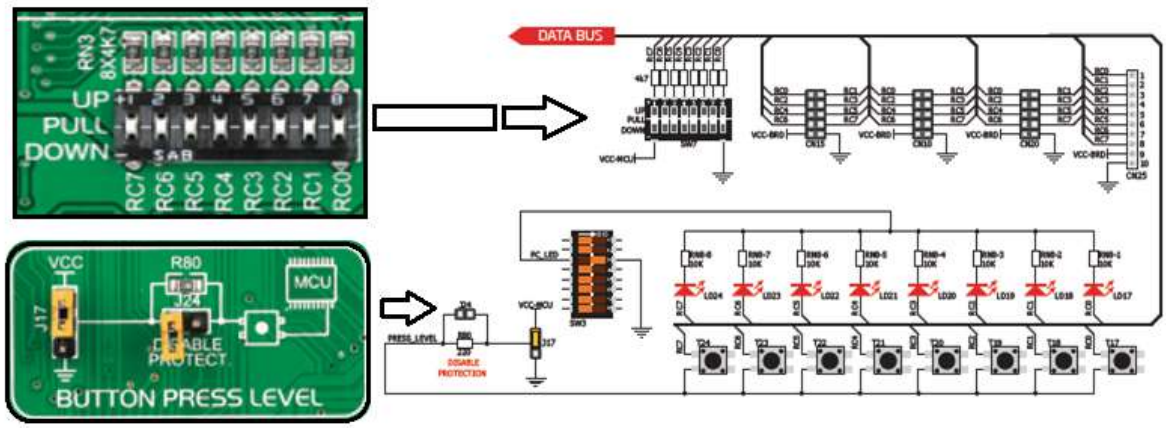


Figure 23. Description de connexions des boutons-poussoirs & LEDs avec le PIC.

B. Étape 2 : Programmation en Mikro C

- Ouvrez le logiciel de développement EasyPIC V7 sur votre ordinateur.
- Créez un nouveau projet en sélectionnant le microcontrôleur approprié pour la carte EasyPIC V7.
- Incluez la bibliothèque EasyPIC V7 nécessaire pour accéder aux fonctions spécifiques de la carte.

```
#include <esypic_v7.h>
```

- Configurez les broches associées aux boutons-poussoirs en mode entrée à l'aide de la fonction TRISx.

C. Étape 3 : Programmation du Contrôle

- Dans la boucle principale, utilisez la fonction **GPIO_Read** pour lire l'état des broches associées aux boutons-poussoirs.
- Utilisez des structures conditionnelles (if-else) pour détecter l'appui de chaque bouton-poussoir.
- Lorsqu'un bouton est appuyé, allumez ou éteignez les 8 LEDs en conséquence en utilisant les fonctions **GPIO_Set** et **GPIO_Clear**.

D. Étape 4 : Programmation en Mikro C - Utilisation de Pull-Up

- Modifiez la configuration des broches des boutons-poussoirs pour utiliser la résistance pull-up interne.

E. Étape 5 : Programmation en Mikro C - Utilisation de Pull-Down

- Modifiez la configuration des broches des boutons-poussoirs pour utiliser la résistance pull-down interne.

II.3.3 Manipulation 3 : Contrôle de la vitesse d'allumage progressif de 8 LEDs par la lecture des 8 boutons-poussoirs

Dans cette manipulation, vous allez travailler sur le contrôle de la vitesse d'allumage progressif de 8 LEDs en fonction de l'appui des 8 boutons-poussoirs correspondants. Chaque bouton-poussoir sera associé à une LED, et en appuyant sur un bouton, la LED correspondante s'allumera progressivement à différentes vitesses. Plusieurs niveaux de vitesse seront définis, et chaque appui sur un bouton-poussoir augmentera la vitesse de clignotement de la LED correspondante.

Identifiez les broches associées aux 8 boutons-poussoirs et aux 8 LEDs sur la carte EasyPIC V7 (LEDs aux broches (C0 à C7) et les boutons-poussoirs aux broches (B0 à B7)).

Configurez les broches associées aux boutons-poussoirs en tant qu'entrées numériques et les broches associées aux LEDs en tant que sorties numériques.

Dans la boucle principale du programme, lisez l'état de chaque bouton-poussoir en utilisant la fonction **GPIO_Read**.

1. Pour chaque bouton-poussoir, associez un compteur de vitesse (une variable) qui déterminera la vitesse d'allumage de la LED correspondante. Initialisez ces compteurs à une vitesse de base.
2. Lorsqu'un bouton-poussoir est appuyé, incrémentez le compteur de vitesse correspondant.
3. Utilisez les compteurs de vitesse pour contrôler la vitesse d'allumage de chaque LED. Plus le compteur est élevé, plus la vitesse d'allumage est rapide.
4. Si le compteur atteint une valeur maximale (correspondant à la vitesse la plus rapide), remettez-le à la vitesse de base.
5. Répétez les étapes 3 à 7 pour chaque bouton-poussoir et LED.

Conseils :

- Utilisez des tableaux pour stocker les états des boutons-poussoirs et les compteurs de vitesse.
- Ajoutez des délais (fonction **Delay_ms()**) pour contrôler la vitesse d'allumage.
- Assurez-vous que les compteurs de vitesse ne dépassent pas leur valeur maximale.

II.3.4 Manipulation 4 : Contrôle de l'Allumage Progressif de 8 LEDs par la Lecture du Potentiomètre

Dans cette manipulation, vous allez explorer la utilisation des entrées analogiques et des sorties numériques sur la carte EasyPIC V7. L'objectif principal est de contrôler l'allumage progressif de 8 LEDs en fonction de la valeur lue à partir du potentiomètre intégré sur la carte. Les étapes de la manipulation sont décrites comme suite :

Étape 1 : Configuration Matérielle

Assurez-vous que la carte EasyPIC V7 est alimentée et prête à être utilisée.

Familiarisez-vous avec les broches associées aux LEDs intégrées.

Identifiez l'emplacement du potentiomètre intégré sur la carte en vous référant à la Figure 24, puis placez les cavaliers dans les emplacements appropriés.

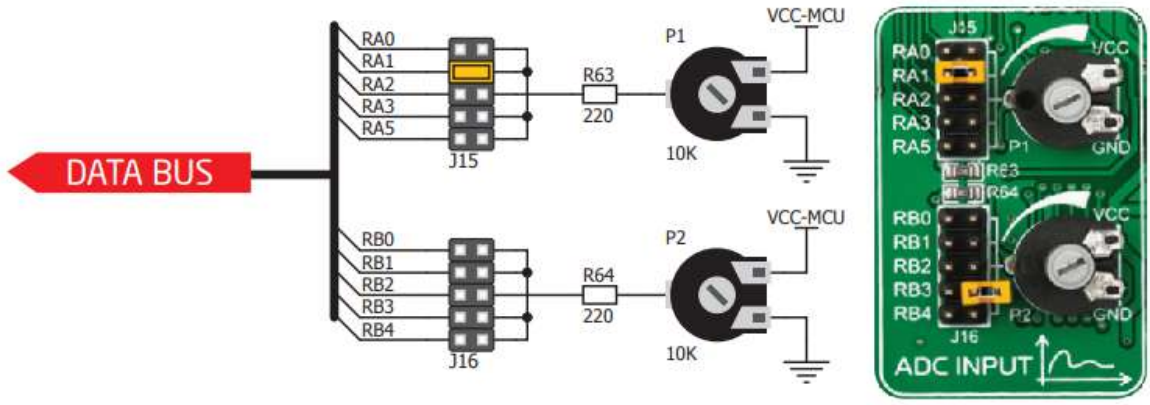


Figure 24. Description de connexions des 2 potentiomètres avec le Bus data de PIC.

A. Étape 2 : Programmation en Mikro C

- Ouvrez l'environnement de développement Mikro C sur votre ordinateur.
- Créez un nouveau projet en sélectionnant le microcontrôleur adapté à la carte EsyPic V7.
- Incluez la bibliothèque EsyPic V7 dans votre projet.

```
#include <easypic_v7.h>
```

- Configurez les broches des LEDs en mode sortie en utilisant la fonction **TRISx**.
- Configurez la broche du potentiomètre en mode entrée en utilisant la fonction **TRISx**.

B. Étape 3 : Programmation du Contrôle

- Créez une boucle infinie pour lire en continu la valeur analogique du potentiomètre en utilisant la fonction **ADC_Read** de la bibliothèque **EasyPIC V7**.
- Divisez la plage de valeurs lues (0 à 1023) en 8 segments égaux. Chaque segment représentera le nombre de LEDs allumées.
- Utilisez une structure conditionnelle (if-else) pour déterminer le nombre de LEDs à allumer en fonction du segment dans lequel se trouve la valeur lue.
- Utilisez les fonctions **GPIO_Set** et **GPIO_Clear** de la bibliothèque **EasyPIC V7** pour allumer et éteindre les LEDs.

Travaux Pratiques N° 3:

Interface Homme-Machine (IHM) avec EsyPIC v7 : Afficheur 7 Segments et LCD128x2

III.1 Objectif

- Configurer l'EasyPIC V7 pour développer une Interface Homme-Machine (IHM) efficace.
- Intégrer et contrôler un afficheur 7 segments pour afficher des données numériques.
- Implémenter la gestion d'un écran LCD128x2 afin d'afficher des informations textuelles claires.

III.2 Introduction

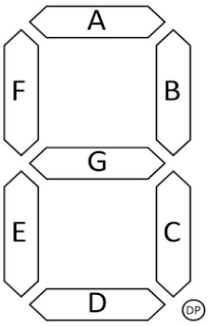
III.2.1 Afficheur 7 segments

Les afficheurs 7 segments sont couramment utilisés pour afficher des nombres et des caractères en utilisant sept segments distincts pour chaque chiffre. Ils offrent une méthode simple pour rendre les informations numériques facilement lisibles. Présents dans divers appareils comme les horloges, compteurs et balances, ces afficheurs sont omniprésents.

Chaque afficheur 7 segments est composé de sept segments individuels disposés de manière à former des motifs numériques ou alphanumériques. Ces segments, étiquetés de "a" à "g", représentent les chiffres de 0 à 9 et parfois des caractères spéciaux. Chacun des segments peut être activé ou désactivé indépendamment, ce qui permet la création de diverses combinaisons visuelles. Le contrôle de ces afficheurs est réalisé au moyen de signaux électriques, généralement fournis par un microcontrôleur. En appliquant la combinaison appropriée de signaux aux segments, il devient possible d'afficher des chiffres individuels et même de simuler des lettres. Dans le tableau ci-dessous, veuillez indiquer les combinaisons correctes pour afficher les chiffres de 0 à 9.

Vous pouvez contrôler les valeurs hexadécimales obtenues grâce a l'un des utilitaires de mikroC. Pour l'afficher : Tools > Seven Segment Editor.

Tableau 1. Codification des chiffres décimaux de 0 à 9 pour l'afficheur 7 segments.



Affichage	Bit 7 DB	Bit 6 G	Bit 5 F	Bit 4 E	Bit 3 D	Bit 2 C	Bit 1 B	Bit 0 A	Hexa
0	0	0	1	1	1	1	1	1	0x3F
1	0								
2	0								
3	0								
4	0								
5	0								
6	0								
7	0								
8	0								
9	0								

III.2.2 Afficheur LCD 128x2

L'afficheur LCD 128x2 est une interface d'affichage graphique qui permet de présenter des informations de manière visuelle et lisible. Contrairement aux afficheurs 7 segments qui affichent principalement des chiffres, l'afficheur LCD 128x2 offre une plus grande flexibilité en termes de contenu affiché, en permettant l'affichage de texte, de symboles et même de graphiques simples. L'afficheur LCD 128x2 se compose d'une matrice de pixels répartis sur une grille de 128 colonnes par 2 lignes. Chaque pixel peut être contrôlé individuellement pour afficher du texte ou des graphiques. Les caractères et les symboles sont généralement générés à l'aide d'un jeu de caractères prédéfini, tandis que les graphiques peuvent être créés en contrôlant les pixels individuels.

III.3 Travail à réaliser

III.3.1 Manipulation 1 : Familiarisation avec les Afficheurs 7 Segments de la Carte EasyPIC V7

Dans cette première manipulation, nous allons explorer les caractéristiques des Afficheurs 7 Segments de la Carte EasyPIC V7. Le schéma des connections des afficheurs est rappelés ci-dessous (figure 25):

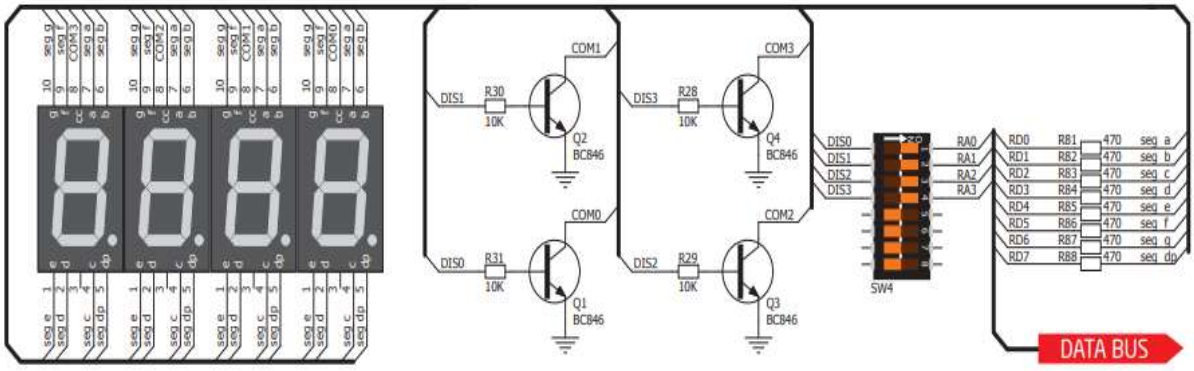


Figure 25. Schéma de connexion des 4 Afficheurs 7 segments avec le bus data de PIC1

1. Quel port permet de sélectionner l'afficheur actif ?
2. Quelles valeurs faut-il envoyer pour sélectionner :
   - L'afficheur DIS0 ?.....
   - L'afficheur DIS1 ?.....
   - L'afficheur DIS2 ?.....
   - L'afficheur DIS3 ?.....
3. Quel port permet d'afficher les segments ?
4. Est-il possible d'activer 2 segments en même temps ?
5. Est-il possible d'activer 2 symboles en même temps ?
6. Quelle est l'utilité du multiplexage ?

Le Tableau 1 représente effectivement une opération de masquage qui sera mise en œuvre au sein d'une fonction du programme MikroC pour convertir chaque symbole à afficher (0, 1, 2, etc.) en un mot binaire. Ce mot binaire doit ensuite être appliqué au port directement lié aux entrées des afficheurs 7 segments de la carte. Par exemple, le symbole 8 (0000 1000b) est substitué par le mot binaire 0111 1111b.

7. En utilisant le Tableau 1, veuillez compléter la fonction de masquage :

```
// Fonction de masquage pour la conversion des symboles en mots binaires
unsigned short  masquerSymbole(unsigned  short  symbole) {
switch(symbole){
  case 0 : return 0x3fF;
  case 1 : return …;
  case 2 : return …;
  case 3 : return …;
```

```
  case 4 : return …;
  case 5 : return …;
  case 6 : return …;
  case 7 : return …;
  case 8 : return …;
  case 9 : return …;
                    }
}
```

8. Substituez les variables x et y par les ports appropriés dans le code principal ci-dessous :

```
Void main() {
TRISx = 0 ;
LATx = 0 ;
TRISy = 0 ;
LATy =0 ;
While(1){ For(i=0 ;i<=9 ;i++) { LATx = 0 ;
                                LATy = masquerSymbole(i);
                                LATx = 1 ;
                               Delay_ms(1000) ;}
          }
}
```

9. Examinez le code. Quelle action pensez-vous que l'afficheur DIS0 accomplira ?
10. Configurez le commutateur SW4 pour autoriser l'activation de l'afficheur DIS0 (position ON de SW4.1).
11. Créez un nouveau projet (TP03/manipulation01) avec une fréquence de 32 MHz.
12. Lors de la configuration du projet (Edit Project), n'oubliez pas d'activer la PLL pour obtenir 32 MHz (8 x 4).
13. Copiez le contenu du fichier manipulation01.c et enregistrez les modifications.
14. Allumez la carte EasyPIC V7.
15. Compilez et programmez (build et program) le microcontrôleur.
16. Vérifiez le bon fonctionnement de votre code.

17. Modifiez le programme pour utiliser successivement l'afficheur DIS1, puis DIS2, et enfin DIS3.

III.3.2 Manipulation 2 : Gestion de deux afficheur 7 segments

Vous êtes à présent sur le point de mettre en action deux afficheurs (DIS0 et DIS1) pour présenter une valeur croissante, de 00 à 90. En raison de la configuration matérielle de la carte EasyPIC V7, il n'est pas possible d'activer simultanément plusieurs afficheurs. Par conséquent, vous devrez réaliser une commutation entre les deux (procédé de multiplexage). Si le multiplexage est exécuté de manière optimale, la transition entre les afficheurs ne devrait pas être perceptible. Dans cette manipulation, vous devrez effectuer un basculement entre les deux afficheurs en suivant l'algorithme décrit ci-dessous.

```
Boucle For (i=1 ;i<=100 ;i++){
Activer DIS0
Afficher les chiffres de poids faible
Tempo (10 ms par exemple)
Activer DIS1
Afficher le chiffre de poids fort
Tempo (10 ms par exemple)
                        }
```

1. Autoriser l'utilisation des afficheurs DIS0 et DIS1 (Switch 4.1 et SW4.2 On)
2. Créez un nouveau projet (TP03/manipulation02) avec une fréquence de 32 MHz.
3. Lors de la configuration du projet (Edit Project), n'oubliez pas d'activer la PLL pour obtenir 32 MHz (8 x 4).
4. En utilisant l'algorithme précédent, rédigez et évaluez un programme qui affiche un compteur de 0 à 90 sur les deux afficheurs. Cette tâche implique l'utilisation des opérateurs de division (**/**) et de modulo (**%**) pour obtenir le résultat souhaité.
5. Pour extraire le chiffre dizaines d'un nombre : ***Nombre_ dizaine = (Nombre/10)%10***
6. Pour extraire le chiffre des unités : ***Nombre_unite = Nombre %10***
7. Modifier la valeur des temporisateurs (100 ms)
8. Quelles observations pouvez-vous faire ? Veuillez fournir une explication.
9. Même avec une valeur de temporisation relativement basse (10 ms ou moins), cette approche de programmation est-elle fiable ?

III.3.3 Manipulation 3 : Réalisation d'un Compteur interactif avec affichage sur les afficheurs 7 segments

Durant cette manipulation, l'objectif réside dans la mise en place d'un compteur interactif, se développant à chaque activation d'un bouton. La valeur courante du compteur doit être clairement affichée sur les afficheurs 7 segments intégrés à la carte EasyPIC V7. Les boutons, judicieusement disposés, comprennent un pour l'augmentation du compteur et un autre pour le réinitialiser. En utilisant un code de programmation rédigé en MikroC, les pressions sur les boutons peuvent être détectées. Il est essentiel de garantir que le compteur évolue avec chaque pression sur le premier bouton, tout en pouvant être remis à zéro grâce au second bouton lorsque nécessaire.

1. Écrivez un programme permettant la mise en place d'un compteur interactif avec Affichage sur les Afficheurs 7 Segments.

III.3.4 Manipulation 4: Exploration de l'Afficheur LCD 128x2 sur la Carte EasyPIC V7

Au cours de cette manipulation, nous allons étudier l'exploitation et le fonctionnement de l'afficheur LCD 128x2 de la Carte EasyPIC V7. Le schéma de connexion de l'afficheur LCD est rappelé ci-dessous (figure 26) :

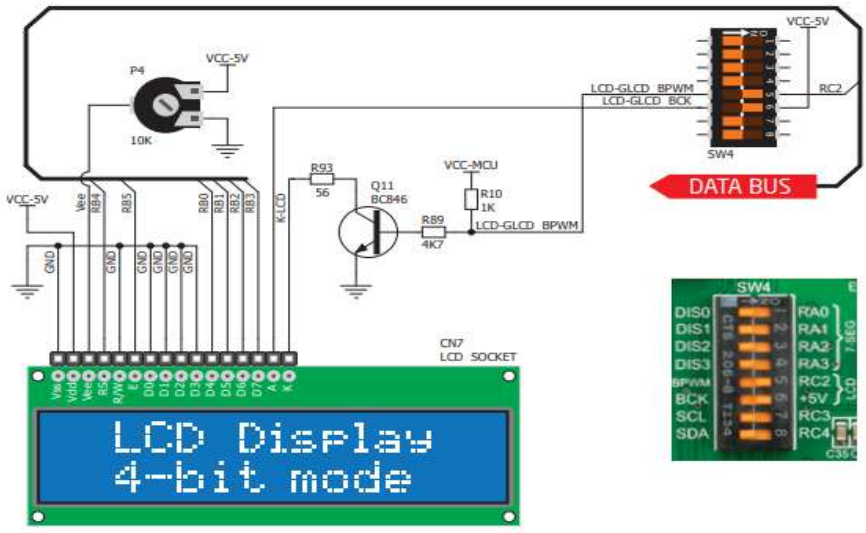


Figure 26. Schéma de connexion d'Afficheur LCD avec le bus data de PIC.

Les broches du LCD 128x2, à savoir D4, D5, D6 et D7, sont liées respectivement aux broches RB0, RB1, RB2 et RB3 des ports B de la carte EasyPIC V7. De manière similaire, les broches Rs et E du LCD sont reliées aux broches RB4 et RB5 du même port B. La broche

Vee est connectée à un potentiomètre afin de permettre le réglage du contraste. Pour activer le LCD, les connecteurs SW4.5 et .6 doivent être en position "On". Ci-dessous se trouve un exemple de programme illustrant comment afficher deux textes, un sur chaque ligne de l'afficheur LCD 128x2. Cet afficheur est directement relié à la carte EasyPIC V 7.

```
// Lcd pinout settings
   sbit LCD_RS at LATB4_bit;  sbit LCD_EN at LATB5_bit;  sbit LCD_D7 at LATB3_bit;
   sbit LCD_D6 at LATB2_bit;  sbit LCD_D5 at LATB1_bit;  sbit LCD_D4 at LATB0_bit;
// Pin direction
   sbit LCD_RS_Direction at TRISB4_bit;  sbit LCD_EN_Direction at TRISB5_bit;
   sbit LCD_D7_Direction at TRISB3_bit;  sbit LCD_D6_Direction at TRISB2_bit;
   sbit LCD_D5_Direction at TRISB1_bit;  sbit LCD_D4_Direction at TRISB0_bit;
char txt1[] = "Travaux pratique N3"; char txt2[] = "EasyPIC7"; char i;
void main(){
  ANSELB = 0;                              // Configurer les broches PORTB comme numériques
  Lcd_Init();                              // Initialiser Lcd
  Lcd_Cmd(_LCD_CLEAR);                     // Effacer l'affichage
  Lcd_Cmd(_LCD_CURSOR_OFF);                // Curseur désactivé
  Lcd_Out(1,6,txt1);                       // Écrit le texte dans la première ligne
  Lcd_Out(2,6,txt2);                       // Écrit le texte dans la deuxième ligne
  Delay_ms(2000);
  Lcd_Cmd(_LCD_CLEAR);                     // Effacer l'affichage
 // Déplacement du texte
  while(1) {
    for(i=0; i<8; i++) {Lcd_Cmd(_LCD_SHIFT_LEFT); // Déplace le texte vers la gauche 7 fois
                  Delay_ms(500);}
    for(i=0; i<8; i++) {Lcd_Cmd(_LCD_SHIFT_RIGHT);// Déplace le texte vers la droite 7 fois
                  Delay_ms(500);}
          }
}
```

1. Examinez le code. Quelle action pensez-vous que l'afficheur LCD 128x2 accomplira ?

III.3.5 Manipulation 5: Conversion Binaire en Décimal avec Affichage sur l'Afficheur LCD

L'objectif de cette manipulation est d'implémenter un programme qui permet de convertir des valeurs binaires en décimal en utilisant les boutons de la carte EasyPIC V7. Les entrées des valeurs binaires seront fournies par l'intermédiaire d'appuis sur les boutons, lesquels sont directement connectés aux ports D de la carte EasyPIC V7. Les valeurs binaires entrées seront ensuite converties en décimal et le résultat sera affiché sur l'afficheur LCD 128x2 de la carte.

Etapes à suivre :

1. Implémentez une logique de programme qui détecte les appuis sur les boutons et enregistre les valeurs binaires correspondantes.
2. Mettez en œuvre un algorithme de conversion binaire en décimal. Ceci peut être réalisé en parcourant les bits binaires et en additionnant les puissances de 2 correspondantes.
3. Affichez le résultat de la conversion décimale sur l'afficheur LCD en utilisant les fonctions appropriées de l'afficheur LCD.
4. Assurez-vous de gérer les éventuelles erreurs, telles que la détection d'appuis incorrects ou la gestion des dépassements de capacité dans la conversion binaire en décimal.
5. Testez votre programme en entrant différentes valeurs binaires à l'aide des boutons et en vérifiant que les résultats de conversion décimale sont correctement affichés sur l'afficheur LCD.
6. Une fois que le programme fonctionne correctement, expérimentez avec différentes valeurs binaires pour valider sa précision et sa robustesse.

Travaux Pratiques N° 4:

Implémentation d'un Thermomètre Numérique avec EsyPIC v7 : Acquisition et Affichage des Données

IV.1 Objectif

- Configurer l'EasyPIC V7 pour créer un thermomètre numérique capable d'acquérir des données de température à partir du DS1820 et du LM35.
- Afficher les résultats de température sur un afficheur LCD.
- Générer un signal sonore avec un buzzer en cas de dépassement du seuil de température.
- Enregistrer les données dans la mémoire EEPROM de la carte EasyPIC V7.

IV.2 Introduction

VII.2.1 Capteur de température numérique DS1820

Le DS1820 est un capteur de température numérique précis opérant sur une plage de -55 à 128°C, utilisant une interface 1-wire® et nécessitant une alimentation de 3V à 5,5V. La communication série 1-wire® permet de connecter plusieurs capteurs sur une seule ligne, avec un code d'identification unique par capteur. EasyPIC V7 propose une prise spécifique (TS1) pour le DS1820, reliée via le cavalier J11.

EasyPIC V7 facilite la communication 1-wire® entre le DS1820 et le microcontrôleur via les broches RA4 ou RE2. Utilisez le cavalier J11 pour choisir entre ces deux lignes. Lors de l'insertion du capteur dans la prise, alignez le demi-cercle sur les repères de la carte avec la partie arrondie du capteur DS1820 pour éviter des dommages. Déconnectez les autres périphériques, LED et résistances de tirage supplémentaires des lignes d'interface pour préserver l'intégrité des signaux et des données. La figure 27 montre le schéma de connexion d'un capteur de température numérique DS1820 ainsi que les quatre modes pour l'activation de capteur. L'explication des modes d'activation de capteur est présente comme suite :

- DS1820 non connecté,
- DS1820 placé dans la prise,
- DS1820 connecté à la broche RE2,
- DS1820 connecté à la broche RA4. Haut du formulaire

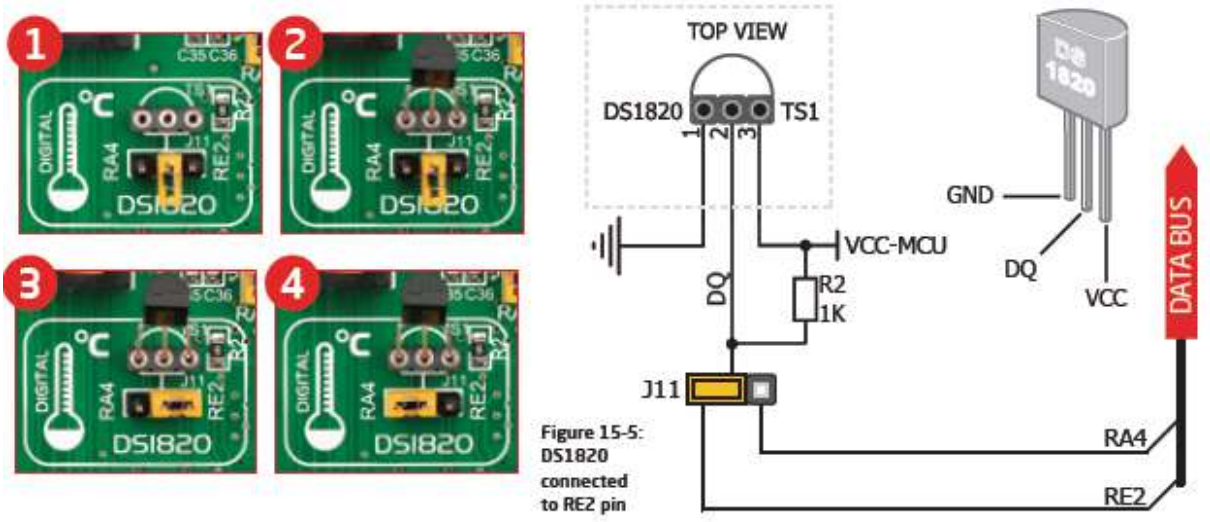


Figure 27. Schéma de connexion d'un capteur de température numérique DS1820 et activation du capteur.

VII.2.2 Capteur de température Analogique - LM35

Le LM35 est un capteur de température à faible coût, en Celsius, offrant un avantage par rapport aux capteurs calibrés en ° Kelvin. Sa faible consommation de courant limite l'auto-échauffement à moins de 0,1°C. EasyPIC V7 permet des lectures analogiques du LM35 de +2°C à +150°C via une prise dédiée (TS2), connectée aux broches microcontrôleur RE2 ou RE1 via le cavalier J25.

EasyPIC V7 permet des lectures analogiques du capteur LM35 via les broches microcontrôleur RE1 ou RE2, sélectionnées avec le cavalier J25. Lors de l'insertion, alignez le demi-cercle sur la carte avec la partie arrondie du capteur LM35 pour éviter des dommages. Une mauvaise connexion peut endommager le capteur de façon permanente. Évitez les interférences en vous assurant qu'aucun autre dispositif n'utilise la ligne analogique pendant les lectures du capteur. La figure 28 montre le schéma de connexion d'un capteur de température analogique TM35 ainsi que les quatre modes pour l'activation de capteur. L'explication des modes d'activation de capteur est présente comme suite :

- LM35 non connecté ;
- LM35 placé dans la prise ;
- LM35 connecté à la broche RE1 ;
- LM35 connecté à la broche RE2.

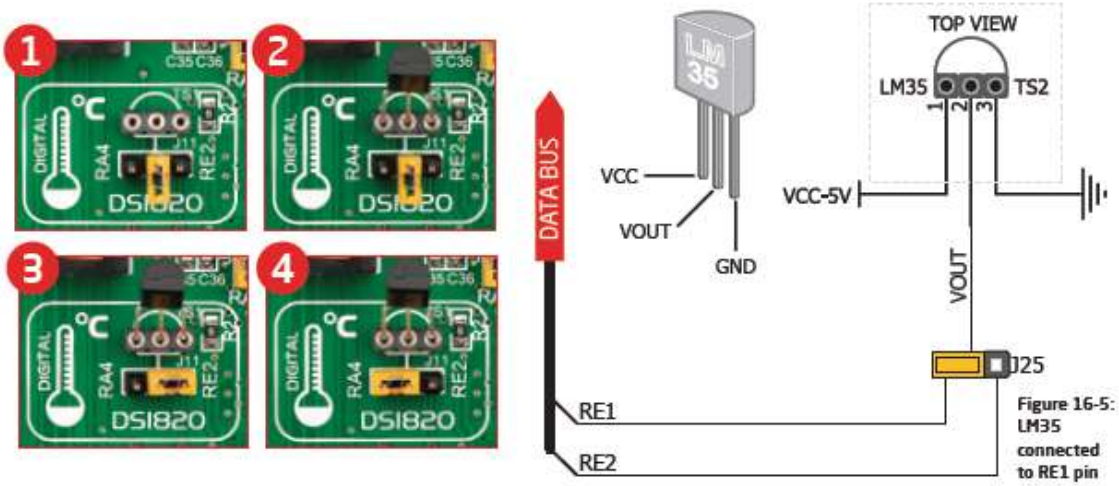


Figure 27. Schéma de connexion d'un capteur de température Analogique TLM35 et activation du capteur.

VII.2.3 Buzzer

Le buzzer est un composant électronique largement utilisé dans de nombreuses applications pour produire des sons et des alertes sonores. C'est un élément essentiel dans les systèmes de signalisation, les alarmes, les dispositifs d'alerte, les jouets électroniques et bien d'autres dispositifs interactifs. Sa capacité à émettre des sons de différentes fréquences et intensités en fait un outil polyvalent pour transmettre des informations auditives à l'utilisateur.

Pour contrôler le buzzer sur l'EasyPIC V7 (figure 29), vous pouvez utiliser les broches de sortie numérique du microcontrôleur, telles que les broches GPIO (General Purpose Input/Output). En programmant le microcontrôleur avec le code approprié, vous pouvez activer ou désactiver le buzzer pour produire des sons selon vos besoins.

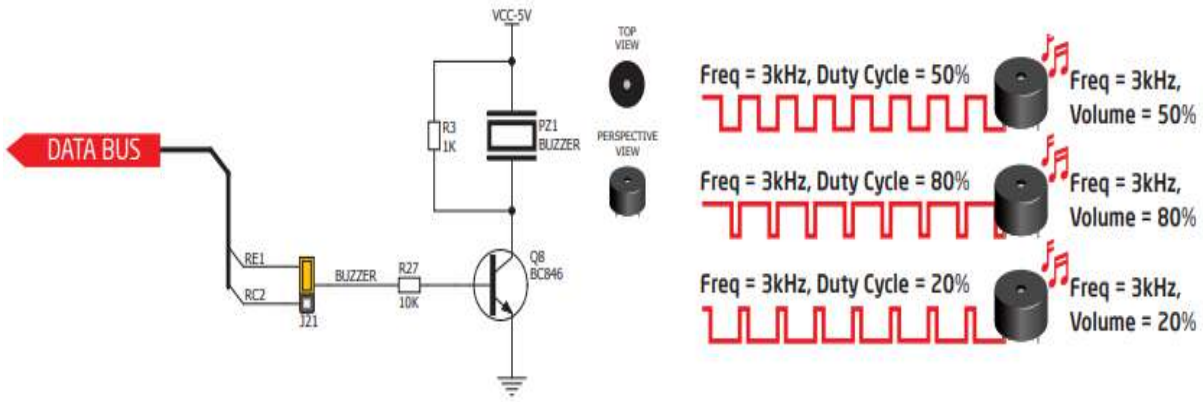


Figure 28. Schéma de connexion de Buzzer piézoélectrique connecté à la broche du microcontrôleur RE1.

VII.3 Travail à réaliser

VII.3.1 Manipulation 1 : Acquisition et Affichage des Températures

Dans cette première manipulation, vous allez explorer l'utilisation des capteurs de température numérique DS1820 et analogique LM35 sur la carte EasyPIC V7. L'objectif principal est de lire la température à partir de ces capteurs et d'afficher les résultats sur l'afficheur LCD intégré à la carte. Les étapes de la manipulation sont décrites comme suit :

1. Connectez correctement les capteurs DS1820 et LM35 à la carte EasyPIC V7.
2. Écrivez un programme MikroC qui lit la température à partir des deux capteurs.
3. Affichez les deux températures (en degrés Celsius) sur l'afficheur LCD avec une mise à jour en temps réel.

VII.3.2 Manipulation 2 : Alarme de Température Élevée avec Buzzer

Dans cette manipulation, nous ajouterons une fonction d'alarme sonore avec un buzzer pour avertir lorsque la température dépasse un seuil prédéfini. Vous intégrerez un buzzer au circuit et programmerez la carte pour générer un signal sonore lorsque la température du capteur LM35 dépasse un seuil spécifié, par exemple, 30°C. Les étapes de la manipulation sont les suivantes :

1. Intégrez un buzzer au circuit et programmez la carte pour générer un signal sonore lorsque la température du capteur LM35 dépasse un seuil prédéfini (par exemple, 30°C).
2. Testez l'alarme en chauffant le capteur LM35 pour simuler une température élevée.
3. Assurez-vous que l'alarme cesse lorsque la température redescend en dessous du seuil.

VII.3.3 Manipulation 3 : Conversion de Température en Degrés Fahrenheit

La troisième manipulation étendra le programme pour inclure une conversion de température en degrés Fahrenheit à partir des valeurs mesurées par le capteur LM35. Vous afficherez simultanément les deux températures (en degrés Celsius et en degrés Fahrenheit) sur l'afficheur LCD en temps réel. Les étapes de la manipulation sont les suivantes :

1. Étendez le programme pour inclure une conversion de température en degrés Fahrenheit à partir des valeurs lues par le capteur LM35.
2. Affichez les deux températures (en degrés Celsius et en degrés Fahrenheit) sur l'afficheur LCD avec une mise à jour en temps réel.
3. Permettez à l'utilisateur de basculer entre les deux unités de température en appuyant sur un bouton (optionnel).

VII.3.4 Manipulation 4: Ajoutez une fonctionnalité pour enregistrer périodiquement les données de température dans la mémoire de la carte EasyPIC V7

Dans cette dernière manipulation, vous allez explorer l'utilisation des entrées analogiques, des sorties numériques, et des capacités de stockage de la carte EasyPIC V7 pour ajouter une fonctionnalité d'enregistrement périodique des données de température. L'objectif principal est de programmer la carte pour enregistrer les données de température (en degrés Celsius ou Fahrenheit) dans sa mémoire à intervalles réguliers. Les étapes de la manipulation sont les suivantes :

1. Programmez la carte pour enregistrer les données de température (en degrés Celsius ou Fahrenheit) dans la mémoire à intervalles réguliers.
2. Créez un mécanisme pour stocker plusieurs valeurs de température et une horodatation correspondante.
3. Affichez les données enregistrées sur l'afficheur LCD ou permettez à l'utilisateur de les télécharger via une interface de communication (UART, USB, etc.) pour une analyse ultérieure.

Liste des abréviations

CAN	Convertisseur Analogique Numérique
DC	Direct Current
E/S	entre / Sortie
EEPROM	Electrically Erasable Programmable Read-Only Memory
FTDI	Future Technology Devices International
Gnd	ground
I/O	Input/Output
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IoT	Internet of Things
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
MISO	Master In Slave Out
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MOSI	Master Out Slave In
PWM	Pulse Width Modulation
RF	radio frequency
RFID	Radio-Frequency Identification
RISC	Reduced Instruction Set Computer
SCK	Serial Clock
SCL	Serial Clock
SDA	Serial Data
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SS	Slave Select
USART	Universal Synchronous & Asynchronous Receiver Transmitter
USB	Universal Serial Bus
Wi-Fi	Wireless Fidelity

Références

Daoud A., Polycopié des TPde la Programmation Graphique du Microcontrôleur PIC16F877A avec le Logiciel Flowcode. 2015

LIONEL, T. Mbiadoun, MARTIN, K. P. E., et PASCAL, Ntsama Eloundou. Universal Module of Acquisition and Transmission of Electrophysiological Signal. *International Journal of Innovative Research in Science, Engineering and Technology*, 2014, vol. 3, no 6, p. 13767-13776.

PICTAYARI LASSAAD, TP 4 Microcontrôleur. Prise en main du kit EasyPic7 et du compilateur mikroc PRO for PIC, 2015

Lefaf, Conception et réalisation du thermomètre électronique, 2015

MEGNAFI, Hicham, ABDELLAOUI, Ghouti, et MR BRAHAMI, Mustapha Anwar. Systèmes à Microcontrôleur. 2019.

ABDELLAOUI, Ghouti et MEGNAFI, Hicham. Systèmes embarqés et temps réel (RaspberryPi). 2019.

BRAHAMI, Mustapha Anwar et MEGNAFI, Hicham. Polycopie TP Réseaux et Protocoles. 2022.

https://www.mikroe.com/easypic-v7

PIC18(L)F2X/4XK22
Data Sheet

28/40/44-Pin, Low-Power, High-Performance Microcontrollers with nanoWatt XLP Technology

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

ISBN: 978-1-60932-175-8

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

MICROCHIP

PIC18(L)F2X/4XK22

28/40/44-Pin, Low-Power, High-Performance Microcontrollers with nanoWatt XLP Technology

High-Performance RISC CPU:

- C Compiler Optimized Architecture:
  - Optional extended instruction set designed to optimize re-entrant code
- Up to 1024 Bytes Data EEPROM
- Up to 64 Kbytes Linear Program Memory Addressing
- Up to 3896 Bytes Linear Data Memory Addressing
- Up to 16 MIPS Operation
- 16-bit Wide Instructions, 8-bit Wide Data Path
- Priority Levels for Interrupts
- 31-Level, Software Accessible Hardware Stack
- 8 x 8 Single-Cycle Hardware Multiplier

Flexible Oscillator Structure:

- Precision 16 MHz Internal Oscillator Block:
  - Factory calibrated to ± 1%
  - Selectable frequencies, 31 kHz to 16 MHz
  - 64 MHz performance available using PLL – no external components required
- Four Crystal modes up to 64 MHz
- Two External Clock modes up to 64 MHz
- 4X Phase Lock Loop (PLL)
- Secondary Oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if peripheral clock stops
  - Two-Speed Oscillator Start-up

Analog Features:

- Analog-to-Digital Converter (ADC) module:
  - 10-bit resolution, up to 30 external channels
  - Auto-acquisition capability
  - Conversion available during Sleep
  - Fixed Voltage Reference (FVR) channel
  - Independent input multiplexing
- Analog Comparator module:
  - Two rail-to-rail analog comparators
  - Independent input multiplexing
- Digital-to-Analog Converter (DAC) module:
  - Fixed Voltage Reference (FVR) with 1.024V, 2.048V and 4.096V output levels
  - 5-bit rail-to-rail resistive DAC with positive and negative reference selection
- Charge Time Measurement Unit (CTMU) module:
  - Supports capacitive touch sensing for touch screens and capacitive switches

Extreme Low-Power Management with nanoWatt XLP:

- Sleep mode: 20 nA, typical
- Watchdog Timer: 300 nA, typical
- Timer1 Oscillator: 800 nA @ 32 kHz
- Peripheral Module Disable

Special Microcontroller Features:

- Full 5.5V Operation – PIC18FXXK22 devices
- 1.8V to 3.6V Operation – PIC18LFXXK22 devices
- Self-Programmable under Software Control
- High/Low-Voltage Detection (HLVD) module:
  - Programmable 16-Level
  - Interrupt on High/Low-Voltage Detection
- Programmable Brown-out Reset (BOR):
  - With software enable option
  - Configurable shutdown in Sleep
- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 131s
- In-Circuit Serial Programming™ (ICSP™):
  - Single-Supply 3V
- In-Circuit Debug (ICD)

Peripheral Highlights:

- Up to 35 I/O Pins plus 1 Input-Only Pin:
  - High-Current Sink/Source 25 mA/25 mA
  - Three programmable external interrupts
  - Four programmable interrupt-on-change
  - Nine programmable weak pull-ups
  - Programmable slew rate
- SR Latch:
  - Multiple Set/Reset input options
- Two Capture/Compare/PWM (CCP) modules
- Three Enhanced CCP (ECCP) modules:
  - One, two or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-Shutdown and Auto-Restart
  - PWM steering
- Two Master Synchronous Serial Port (MSSP) modules:
  - 3-wire SPI (supports all 4 modes)
  - I^2C™ Master and Slave modes with address mask

- Two Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) modules:
  - Supports RS-485, RS-232 and LIN
  - RS-232 operation using internal oscillator
  - Auto-Wake-up on Break
  - Auto-Baud Detect

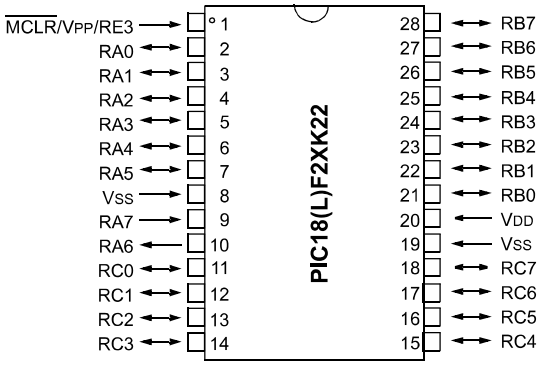
Device	Program Memory		Data Memory		I/O[1]	10-bit A/D Channels[2]	CCP	ECCP (Full-Bridge)	ECCP (Half-Bridge)	MSSP		EUSART	Comparator	CTMU	BOR/LVD	SR Latch	8-bit Timer	16-bit Timer
	Flash (Bytes)	# Single-Word Instructions	SRAM (Bytes)	EEPROM (Bytes)						SPI	I²C™							
PIC18(L)F23K22	8K	4096	512	256	25	19	2	1	2	2	2	2	2	Y	Y	Y	3	4
PIC18(L)F24K22	16K	8192	768	256	25	19	2	1	2	2	2	2	2	Y	Y	Y	3	4
PIC18(L)F25K22	32K	16384	1536	256	25	19	2	1	2	2	2	2	2	Y	Y	Y	3	4
PIC18(L)F26K22	64k	32768	3896	1024	25	19	2	1	2	2	2	2	2	Y	Y	Y	3	4
PIC18(L)F43K22	8K	4096	512	256	36	30	2	2	1	2	2	2	2	Y	Y	Y	3	4
PIC18(L)F44K22	16K	8192	768	256	36	30	2	2	1	2	2	2	2	Y	Y	Y	3	4
PIC18(L)F45K22	32K	16384	1536	256	36	30	2	2	1	2	2	2	2	Y	Y	Y	3	4
PIC18(L)F46K22	64k	32768	3896	1024	36	30	2	2	1	2	2	2	2	Y	Y	Y	3	4

Note 1: One pin is input only.

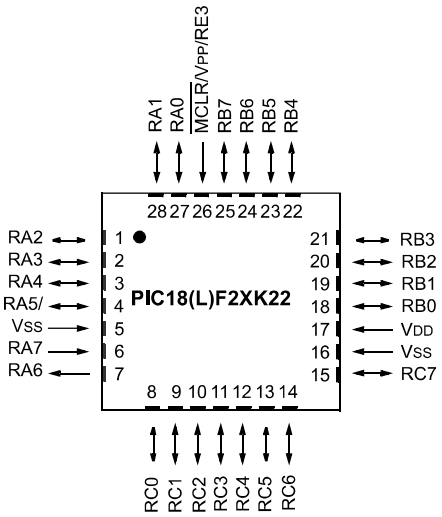
2: Channel count includes internal FVR and DAC channels.

Pin Diagrams

28-pin PDIP, SOIC, SSOP



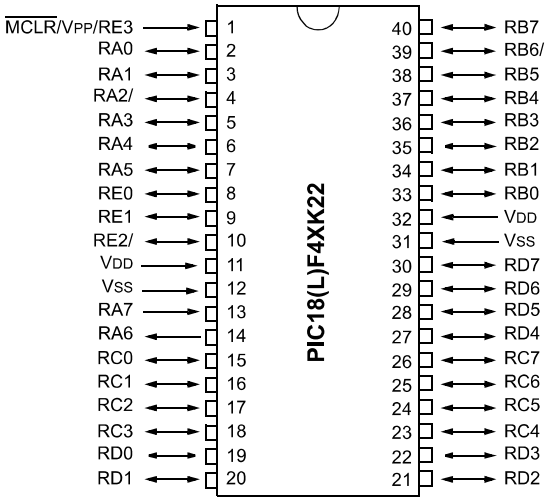
28-pin QFN, UQFN(1)



Note 1: The 28-pin UQFN package is available only for PIC18(L)F23K22 and PIC18(L)F24K22.

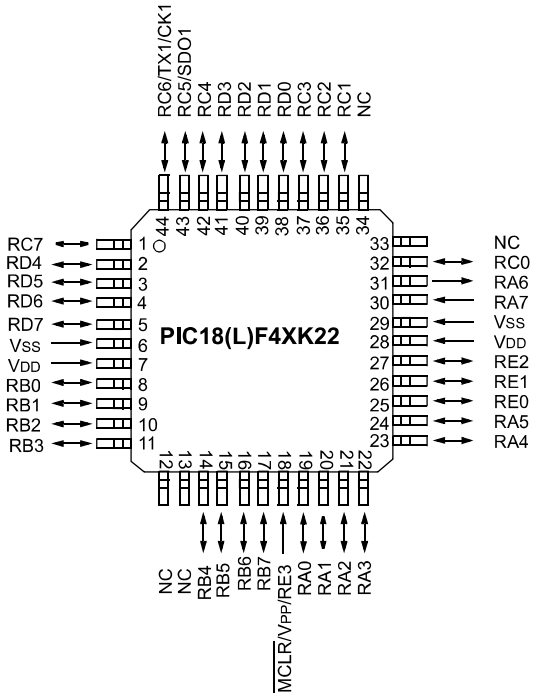
Pin Diagrams

40-pin PDIP



Pin Diagrams (Cont.'d)

44-pin TQFP



44-pin QFN

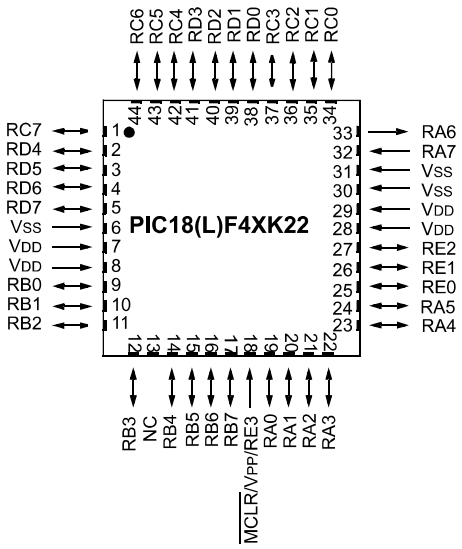


TABLE 1: PIC18(L)F2XK22 PIN SUMMARY

28-SSOP, SOIC 28-SPDIP	28-QFN, UQFN	I/O	Analog	Comparator	CTMU	SR Latch	Reference	(E)CCP	EUSART	MSSP	Timers	Interrupts	Pull-up	Basic
2	27	RA0	AN0	C12IN0-										
3	28	RA1	AN1	C12IN1-										
4	1	RA2	AN2	C2IN+			VREF-/ DACOUT							
5	2	RA3	AN3	C1IN+			VREF+							
6	3	RA4		C1OUT		SRQ		CCP5			T0CKI			
7	4	RA5	AN4	C2OUT		SRNQ	HLVDIN			$\overline{SS1}$				
10	7	RA6												OSC2/ CLKO
9	6	RA7												OSC1/ CLKI
21	18	RB0	AN12			SRI		CCP4 FLT0		$\overline{SS2}$		INT0	Y	
22	19	RB1	AN10	C12IN3-				P1C		SCK2/ SCL2		INT1	Y	
23	20	RB2	AN8		CTED1			P1B		SDI2/ SDA		INT2	Y	
24	21	RB3	AN9	C12IN2-	CTED2			CCP2/ P2A(1)		SDO2			Y	
25	22	RB4	AN11					P1D			T5G	IOC	Y	
26	23	RB5	AN13					CCP3/ P3A(4) P2B(3)			T1G T3CKI(2)	IOC	Y	
27	24	RB6							TX2/CK2			IOC	Y	PGC
28	25	RB7							RX2/DT2			IOC	Y	PGD
11	8	RC0						P2B(3)			SOSCO/ T1CKI T3CKI(2) T3G			
12	9	RC1						CCP2/ P2A(1)			SOSCI			
13	10	RC2	AN14		CTPLS			CCP1/ P1A			T5CKI			
14	11	RC3	AN15							SCK1/ SCL1				
15	12	RC4	AN16							SDI1/ SDA1				
16	13	RC5	AN17							SDO1				
17	14	RC6	AN18					CCP3/ P3A(4)	TX1/CK1					
18	15	RC7	AN19					P3B	RX1/DT1					
1	26	RE3												$\overline{MCLR}$/ VPP
8	5													VSS
19	16													VSS
20	17													VDD

Note 1: CCP2/P2A multiplexed in fuses.
2: T3CKI multiplexed in fuses.
3: P2B multiplexed in fuses.
4: CCP3/P3A multiplexed in fuses.

TABLE 2: PIC18(L)F4XK22 PIN SUMMARY

40-PDIP	44-TQFP	44-QFN	I/O	Analog	Comparator	CTMU	SR Latch	Reference	(E)CCP	EUSART	MSSP	Timers	Interrupts	Pull-up	Basic
2	19	19	RA0	AN0	C12IN0-										
3	20	20	RA1	AN1	C12IN1-										
4	21	21	RA2	AN2	C2IN+			VREF- DACOUT							
5	22	22	RA3	AN3	C1IN+			VREF+							
6	23	23	RA4		C1OUT		SRQ					T0CKI			
7	24	24	RA5	AN4	C2OUT		SRNQ	HLVDIN			$\overline{SS1}$				
14	31	33	RA6												OSC2/ CLKO
13	30	32	RA7												OSC1/ CLKI
33	8	9	RB0	AN12			SRI		FLT0				INT0	Y	
34	9	10	RB1	AN10	C12IN3-								INT1	Y	
35	10	11	RB2	AN8		CTED1							INT2	Y	
36	11	12	RB3	AN9	C12IN2-	CTED2			CCP2/ P2A(1)					Y	
37	14	14	RB4	AN11								T5G	IOC	Y	
38	15	15	RB5	AN13					CCP3/ P3A(4)			T1G T3CKI(2)	IOC	Y	
39	16	16	RB6										IOC	Y	PGC
40	17	17	RB7										IOC	Y	PGD
15	32	34	RC0						P2B(5)			SOSCO/ T1CKI T3CKI(2) T3G			
16	35	35	RC1						CCP2(1) P2A			SOSCI			
17	36	36	RC2	AN14		CTPLS			CCP1/ P1A			T5CKI			
18	37	37	RC3	AN15							SCK1/ SCL1				
23	42	42	RC4	AN16							SDI1/ SDA1				
24	43	43	RC5	AN17							SDO1				
25	44	44	RC6	AN18						TX1/ CK1					
26	1	1	RC7	AN19						RX1/ DT1					
19	38	38	RD0	AN20							SCK2/ SCL2				
20	39	39	RD1	AN21					CCP4		SDI2/ SDA2				
21	40	40	RD2	AN22					P2B(5)						
22	41	41	RD3	AN23					P2C		$\overline{SS2}$				
27	2	2	RD4	AN24					P2D		SD02				
28	3	3	RD5	AN25					P1B						
29	4	4	RD6	AN26					P1C	TX2 CK2					
30	5	5	RD7	AN27					P1D	RX2/ DT2					
8	25	25	RE0	AN5					CCP3/ P3A(4)						

Note 1: CCP2 multiplexed in fuses.
2: T3CKI multiplexed in fuses.
3: Pins are enabled on -ICE derivative only, otherwise they are No Connects.
4: CCP3/P3A multiplexed in fuses.
5: P2B multiplexed in fuses.

TABLE 2: PIC18(L)F4XK22 PIN SUMMARY (CONTINUED)

40-PDIP	44-TQFP	44-QFN	I/O	Analog	Comparator	CTMU	SR Latch	Reference	(E)CCP	EUSART	MSSP	Timers	Interrupts	Pull-up	Basic
9	26	26	RE1	AN6					P3B						
10	27	27	RE2	AN7					CCP5						
1	18	18	RE3											Y	$\overline{\text{MCLR}}$/ V_{PP}
11	7	7,8													V_{DD}
32	28	28, 29													V_{DD}
12	6	6													V_{SS}
31	29	30, 31													V_{SS}
—	12(3)	—													
—	13(3)	—													
—	33(3)	—													
—	34	13	NC												

Note 1: CCP2 multiplexed in fuses.
2: T3CKI multiplexed in fuses.
3: Pins are enabled on -ICE derivative only, otherwise they are No Connects.
4: CCP3/P3A multiplexed in fuses.
5: P2B multiplexed in fuses.

TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@mail.microchip.com** or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (480) 792-7277

When contacting a sales office or the literature center, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

Register on our web site at **www.microchip.com/cn** to receive the most current information on all of our products.

1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F23K22
- PIC18LF23K22
- PIC18F24K22
- PIC18LF24K22
- PIC18F25K22
- PIC18LF25K22
- PIC18F26K22
- PIC18LF26K22
- PIC18F43K22
- PIC18LF43K22
- PIC18F44K22
- PIC18LF44K22
- PIC18F45K22
- PIC18LF45K22
- PIC18F46K22
- PIC18LF46K22

This family offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high-endurance, Flash program memory. On top of these features, the PIC18(L)F2X/4XK22 family introduces design enhancements that make these microcontrollers a logical choice for many high-performance, power sensitive applications.

1.1 New Core Features

1.1.1 nanoWatt TECHNOLOGY

All of the devices in the PIC18(L)F2X/4XK22 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Alternate Run Modes:** By clocking the controller from the Timer1 source or the internal oscillator block, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operation requirements.
- **On-the-fly Mode Switching:** The power-managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application's software design.
- **Low Consumption in Key Modules:** The power requirements for both Timer1 and the Watchdog Timer are minimized. See **Section 27.0 "Electrical Characteristics"** for values.

1.1.2 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18(L)F2X/4XK22 family offer ten different oscillator options, allowing users a wide range of choices in developing application hardware. These include:

- Four Crystal modes, using crystals or ceramic resonators
- Two External Clock modes, offering the option of using two pins (oscillator input and a divide-by-4 clock output) or one pin (oscillator input, with the second pin reassigned as general I/O)
- Two External RC Oscillator modes with the same pin options as the External Clock modes
- An internal oscillator block which contains a 16 MHz HFINTOSC oscillator and a 31 kHz LFINTOSC oscillator, which together provide 8 user selectable clock frequencies, from 31 kHz to 16 MHz. This option frees the two oscillator pins for use as additional general purpose I/O.
- A Phase Lock Loop (PLL) frequency multiplier, available to both external and internal oscillator modes, which allows clock speeds of up to 64 MHz. Used with the internal oscillator, the PLL gives users a complete selection of clock speeds, from 31 kHz to 64 MHz – all without using an external crystal or clock circuit.

Besides its availability as a clock source, the internal oscillator block provides a stable reference source that gives the family additional features for robust operation:

- **Fail-Safe Clock Monitor:** This option constantly monitors the main clock source against a reference signal provided by the LFINTOSC. If a clock failure occurs, the controller is switched to the internal oscillator block, allowing for continued operation or a safe application shutdown.
- **Two-Speed Start-up:** This option allows the internal oscillator to serve as the clock source from Power-on Reset, or wake-up from Sleep mode, until the primary clock source is available.

1.2 Other Special Features

- **Memory Endurance:** The Flash cells for both program memory and data EEPROM are rated to last for many thousands of erase/write cycles – up to 10K for program memory and 100K for EEPROM. Data retention without refresh is conservatively estimated to be greater than 40 years.
- **Self-programmability:** These devices can write to their own program memory spaces under internal software control. By using a bootloader routine located in the protected Boot Block at the top of program memory, it becomes possible to create an application that can update itself in the field.
- **Extended Instruction Set:** The PIC18(L)F2X/4XK22 family introduces an optional extension to the PIC18 instruction set, which adds 8 new instructions and an Indexed Addressing mode. This extension, enabled as a device configuration option, has been specifically designed to optimize re-entrant application code originally developed in high-level languages, such as C.
- **Enhanced CCP module:** In PWM mode, this module provides 1, 2 or 4 modulated outputs for controlling half-bridge and full-bridge drivers. Other features include:
  - Auto-Shutdown, for disabling PWM outputs on interrupt or other select conditions
  - Auto-Restart, to reactivate outputs once the condition has cleared
  - Output steering to selectively enable one or more of 4 outputs to provide the PWM signal.
- **Enhanced Addressable EUSART:** This serial communication module is capable of standard RS-232 operation and provides support for the LIN bus protocol. Other enhancements include automatic baud rate detection and a 16-bit Baud Rate Generator for improved resolution. When the microcontroller is using the internal oscillator block, the EUSART provides stable operation for applications that talk to the outside world without using an external crystal (or its accompanying power requirement).
- **10-bit A/D Converter:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period and thus, reduce code overhead.
- **Extended Watchdog Timer (WDT):** This enhanced version incorporates a 16-bit postscaler, allowing an extended time-out range that is stable across operating voltage and temperature. See **Section 27.0 "Electrical Characteristics"** for time-out periods.
- **Charge Time Measurement Unit (CTMU)**
- **SR Latch Output:**

1.3 Details on Individual Family Members

Devices in the PIC18(L)F2X/4XK22 family are available in 28-pin and 40/44-pin packages. The block diagram for the device family is shown in Figure 1-1.

The devices have the following differences:

1. Flash program memory
2. Data Memory SRAM
3. Data Memory EEPROM
4. A/D channels
5. I/O ports
6. ECCP modules (Full/Half Bridge)
7. Input Voltage Range/Power Consumption

All other features for devices in this family are identical. These are summarized in Table 1-1.

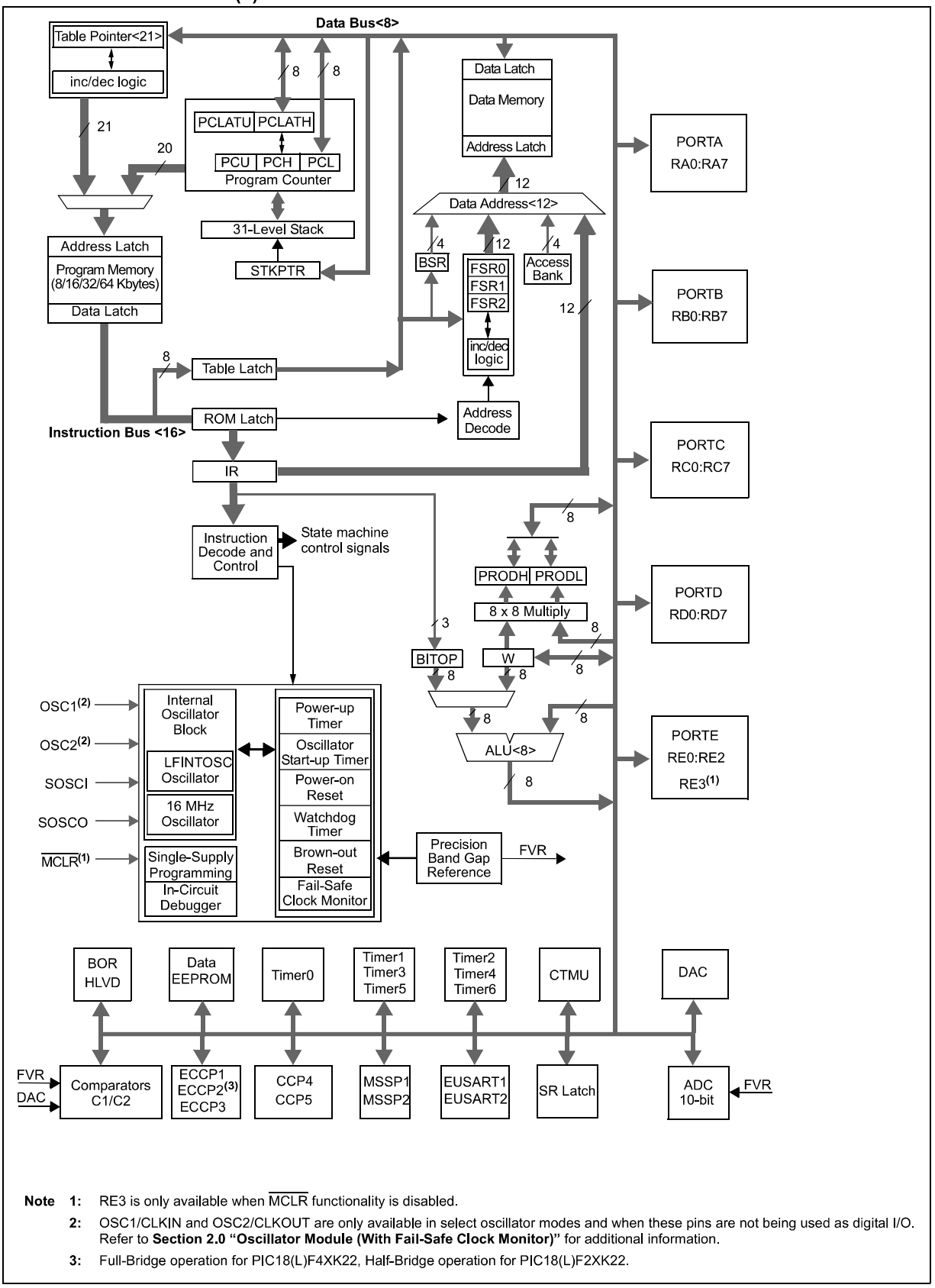
The pinouts for all devices are listed in the pin summary tables: Table 1 and Table 2, and I/O description tables: Table 1-2 and Table 1-3.

TABLE 1-1: DEVICE FEATURES

Features	PIC18F23K22 PIC18LF23K22	PIC18F24K22 PIC18LF24K22	PIC18F25K22 PIC18LF25K22	PIC18F26K22 PIC18LF26K22	PIC18F43K22 PIC18LF43K22	PIC18F44K22 PIC18LF44K22	PIC18F45K22 PIC18LF45K22	PIC18F46K22 PIC18LF46K22
Program Memory (Bytes)	8192	16384	32768	65536	8192	16384	32768	65536
Program Memory (Instructions)	4096	8192	16384	32768	4096	8192	16384	32768
Data Memory (Bytes)	512	768	1536	3896	512	768	1536	3896
Data EEPROM Memory (Bytes)	256	256	256	1024	256	256	256	1024
I/O Ports	A, B, C, E(1)	A, B, C, E(1)	A, B, C, E(1)	A, B, C, E(1)	A, B, C, D, E	A, B, C, D, E	A, B, C, D, E	A, B, C, D, E
Capture/Compare/PWM Modules (CCP)	2	2	2	2	2	2	2	2
Enhanced CCP Modules (ECCP) - Half Bridge	2	2	2	2	1	1	1	1
Enhanced CCP Modules (ECCP) - Full Bridge	1	1	1	1	2	2	2	2
10-bit Analog-to-Digital Module (ADC)	2 internal 17 input	2 internal 17 input	2 internal 17 input	2 internal 17 input	2 internal 28 input	2 internal 28 input	2 internal 28 input	2 internal 28 input
Packages	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN 28-pin UQFN	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN 28-pin UQFN	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin QFN 44-pin TQFP	40-pin PDIP 44-pin QFN 44-pin TQFP	40-pin PDIP 44-pin QFN 44-pin TQFP	40-pin PDIP 44-pin QFN 44-pin TQFP
Interrupt Sources	33							
Timers (16-bit)	4							
Serial Communications	2 MSSP, 2 EUSART							
SR Latch	Yes							
Charge Time Measurement Unit Module (CTMU)	Yes							
Programmable High/Low-Voltage Detect (HLVD)	Yes							
Programmable Brown-out Reset (BOR)	Yes							
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Overflow, Stack Underflow (PWRT, OST), MCLR, WDT							
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled							
Operating Frequency	DC - 64 MHz							

Note 1: PORTE contains the single RE3 read-only bit.

FIGURE 1-1: PIC18(L)F2X/4XK22 FAMILY BLOCK DIAGRAM



Note 1: RE3 is only available when $\overline{\text{MCLR}}$ functionality is disabled.

2: OSC1/CLKIN and OSC2/CLKOUT are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Module (With Fail-Safe Clock Monitor)"** for additional information.

3: Full-Bridge operation for PIC18(L)F4XK22, Half-Bridge operation for PIC18(L)F2XK22.

TABLE 1-2: PIC18(L)F2XK22 PINOUT I/O DESCRIPTIONS

Pin Number		Pin Name	Pin Type	Buffer Type	Description
PDIP, SOIC	QFN				
2	27	RA0/C12IN0-/AN0			
		RA0	I/O	TTL	Digital I/O.
		C12IN0-	I	Analog	Comparators C1 and C2 inverting input.
		AN0	I	Analog	Analog input 0.
3	28	RA1/C12IN1-/AN1			
		RA1	I/O	TTL	Digital I/O.
		C12IN1-	I	Analog	Comparators C1 and C2 inverting input.
		AN1	I	Analog	Analog input 1.
4	1	RA2/C2IN+/AN2/DACOUT/VREF-			
		RA2	I/O	TTL	Digital I/O.
		C2IN+	I	Analog	Comparator C2 non-inverting input.
		AN2	I	Analog	Analog input 2.
		DACOUT	O	Analog	DAC Reference output.
		VREF-	I	Analog	A/D reference voltage (low) input.
5	2	RA3/C1IN+/AN3/VREF+			
		RA3	I/O	TTL	Digital I/O.
		C1IN+	I	Analog	Comparator C1 non-inverting input.
		AN3	I	Analog	Analog input 3.
		VREF+	I	Analog	A/D reference voltage (high) input.
6	3	RA4/CCP5/C1OUT/SRQ/T0CKI			
		RA4	I/O	TTL	Digital I/O.
		CCP5	I/O	ST	Capture 5 input/Compare 5 output/PWM 5 output.
		C1OUT	O	CMOS	Comparator C1 output.
		SRQ	O	TTL	SR Latch Q output.
		T0CKI	I	ST	Timer0 external clock input.
7	4	RA5/C2OUT/SRNQ/$\overline{SS1}$/HLVDIN/AN4			
		RA5	I/O	TTL	Digital I/O.
		C2OUT	O	CMOS	Comparator C2 output.
		SRNQ	O	TTL	SR Latch $\overline{Q}$ output.
		$\overline{SS1}$	I	TTL	SPI slave select input (MSSP1).
		HLVDIN	I	Analog	High/Low-Voltage Detect input.
		AN4	I	Analog	Analog input 4.
10	7	RA6/CLKO/OSC2			
		RA6	I/O	TTL	Digital I/O.
		CLKO	O		In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
		OSC2	O		Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power.

Note 1: Default pin assignment for P2B, T3CKI, CCP3 and CCP2 when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are set.

2: Alternate pin assignment for P2B, T3CKI, CCP3 and CCP2 when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are clear.

TABLE 1-2: PIC18(L)F2XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Number		Pin Name	Pin Type	Buffer Type	Description
PDIP, SOIC	QFN				
9	6	RA7/CLKI/OSC1			
		RA7	I/O	TTL	Digital I/O.
		CLKI	I	CMOS	External clock source input. Always associated with pin function OSC1.
		OSC1	I	ST	Oscillator crystal input or external clock source input ST buffer when configured in RC mode; CMOS otherwise.
21	18	RB0/INT0/CCP4/FLT0/SRI/$\overline{SS2}$/AN12			
		RB0	I/O	TTL	Digital I/O.
		INT0	I	ST	External interrupt 0.
		CCP4	I/O	ST	Capture 4 input/Compare 4 output/PWM 4 output.
		FLT0	I	ST	PWM Fault input for ECCP Auto-Shutdown.
		SRI	I	ST	SR Latch input.
		$\overline{SS2}$	I	TTL	SPI slave select input (MSSP2).
		AN12	I	Analog	Analog input 12.
22	19	RB1/INT1/P1C/SCK2/SCL2/C12IN3-/AN10			
		RB1	I/O	TTL	Digital I/O.
		INT1	I	ST	External interrupt 1.
		P1C	O	CMOS	Enhanced CCP1 PWM output.
		SCK2	I/O	ST	Synchronous serial clock input/output for SPI mode (MSSP2).
		SCL2	I/O	ST	Synchronous serial clock input/output for I^2C™ mode (MSSP2).
		C12IN3-	I	Analog	Comparators C1 and C2 inverting input.
		AN10	I	Analog	Analog input 10.
23	20	RB2/INT2/CTED1/P1B/SDI2/SDA2/AN8			
		RB2	I/O	TTL	Digital I/O.
		INT2	I	ST	External interrupt 2.
		CTED1	I	ST	CTMU Edge 1 input.
		P1B	O	CMOS	Enhanced CCP1 PWM output.
		SDI2	I	ST	SPI data in (MSSP2).
		SDA2	I/O	ST	I^2C™ data I/O (MSSP2).
		AN8	I	Analog	Analog input 8.
24	21	RB3/CTED2/P2A/CCP2/SDO2/C12IN2-/AN9			
		RB3	I/O	TTL	Digital I/O.
		CTED2	I	ST	CTMU Edge 2 input.
		P2A	O	CMOS	Enhanced CCP2 PWM output.
		CCP2(2)	I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
		SDO2	O	—	SPI data out (MSSP2).
		C12IN2-	I	Analog	Comparators C1 and C2 inverting input.
		AN9	I	Analog	Analog input 9.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power.

Note 1: Default pin assignment for P2B, T3CKI, CCP3 and CCP2 when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are set.

2: Alternate pin assignment for P2B, T3CKI, CCP3 and CCP2 when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are clear.

TABLE 1-2: PIC18(L)F2XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Number		Pin Name	Pin Type	Buffer Type	Description
PDIP, SOIC	QFN				
25	22	RB4/IOC0/P1D/T5G/AN11			
		RB4	I/O	TTL	Digital I/O.
		IOC0	I	TTL	Interrupt-on-change pin.
		P1D	O	CMOS	Enhanced CCP1 PWM output.
		T5G	I	ST	Timer5 external clock gate input.
		AN11	I	Analog	Analog input 11.
26	23	RB5/IOC1/P2B/P3A/CCP3/T3CKI/T1G/AN13			
		RB5	I/O	TTL	Digital I/O.
		IOC1	I	TTL	Interrupt-on-change pin.
		P2B(1)	O	CMOS	Enhanced CCP2 PWM output.
		P3A(1)	O	CMOS	Enhanced CCP3 PWM output.
		CCP3(1)	I/O	ST	Capture 3 input/Compare 3 output/PWM 3 output.
		T3CKI(2)	I	ST	Timer3 clock input.
		T1G	I	ST	Timer1 external clock gate input.
		AN13	I	Analog	Analog input 13.
27	24	RB6/IOC2/TX2/CK2/PGC			
		RB6	I/O	TTL	Digital I/O.
		IOC2	I	TTL	Interrupt-on-change pin.
		TX2	O	—	EUSART 2 asynchronous transmit.
		CK2	I/O	ST	EUSART 2 synchronous clock (see related RXx/DTx).
		PGC	I/O	ST	In-Circuit Debugger and ICSP™ programming clock pin.
28	25	RB7/IOC3/RX2/DT2/PGD			
		RB7	I/O	TTL	Digital I/O.
		IOC3	I	TTL	Interrupt-on-change pin.
		RX2	I	ST	EUSART 2 asynchronous receive.
		DT2	I/O	ST	EUSART 2 synchronous data (see related TXx/CKx).
		PGD	I/O	ST	In-Circuit Debugger and ICSP™ programming data pin.
11	8	RC0/P2B/T3CKI/T3G/T1CKI/SOSCO			
		RC0	I/O	TTL	Digital I/O.
		P2B(2)	O	CMOS	Enhanced CCP1 PWM output.
		T3CKI(1)	I	ST	Timer3 clock input.
		T3G	I	ST	Timer3 external clock gate input.
		T1CKI	I	ST	Timer1 clock input.
		SOSCO	O	—	Secondary oscillator output.
12	9	RC1/P2A/CCP2/SOSCI			
		RC1	I/O	TTL	Digital I/O.
		P2A	O	CMOS	Enhanced CCP2 PWM output.
		CCP2(1)	I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
		SOSCI	I	Analog	Secondary oscillator input.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power.

Note 1: Default pin assignment for P2B, T3CKI, CCP3 and CCP2 when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are set.

2: Alternate pin assignment for P2B, T3CKI, CCP3 and CCP2 when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are clear.

TABLE 1-2: PIC18(L)F2XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Number		Pin Name	Pin Type	Buffer Type	Description
PDIP, SOIC	QFN				
13	10	RC2/CTPLS/P1A/CCP1/T5CKI/AN14			
		RC2	I/O	TTL	Digital I/O.
		CTPLS	O	—	CTMU pulse generator output.
		P1A	O	CMOS	Enhanced CCP1 PWM output.
		CCP1	I/O	ST	Capture 1 input/Compare 1 output/PWM 1 output.
		T5CKI	I	ST	Timer5 clock input.
		AN14	I	Analog	Analog input 14.
14	11	RC3/SCK1/SCL1/AN15			
		RC3	I/O	TTL	Digital I/O.
		SCK1	I/O	ST	Synchronous serial clock input/output for SPI mode (MSSP2).
		SCL1	I/O	ST	Synchronous serial clock input/output for I^2C™ mode (MSSP2).
		AN15	I	Analog	Analog input 15.
15	12	RC4/SDI1/SDA1/AN16			
		RC4	I/O	TTL	Digital I/O.
		SDI1	I	ST	SPI data in (MSSP1).
		SDA1	I/O	ST	I^2C™ data I/O (MSSP1).
		AN16	I	Analog	Analog input 16.
16	13	RC5/SDO1/AN17			
		RC5	I/O	TTL	Digital I/O.
		SDO1	O	—	SPI data out (MSSP1).
		AN17	I	Analog	Analog input 17.
17	14	RC6/P3A/CCP3/TX1/CK1/AN18			
		RC6	I/O	TTL	Digital I/O.
		P3A(2)	O	CMOS	Enhanced CCP3 PWM output.
		CCP3(2)	I/O	ST	Capture 3 input/Compare 3 output/PWM 3 output.
		TX1	O	—	EUSART 1 asynchronous transmit.
		CK1	I/O	ST	EUSART 1 synchronous clock (see related RXx/DTx).
		AN18	I	Analog	Analog input 18.
18	15	RC7/P3B/RX1/DT1/AN19			
		RC7	I/O	TTL	Digital I/O.
		P3B	O	CMOS	Enhanced CCP3 PWM output.
		RX1	I	ST	EUSART 1 asynchronous receive.
		DT1	I/O	ST	EUSART 1 synchronous data (see related TXx/CKx).
		AN19	I	Analog	Analog input 19.
1	26	RE3/VPP/$\overline{\text{MCLR}}$			
		RE3	I	ST	Digital input.
		VPP	P		Programming voltage input.
		$\overline{\text{MCLR}}$	I	ST	Active-Low Master Clear (device Reset) input.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power.

Note 1: Default pin assignment for P2B, T3CKI, CCP3 and CCP2 when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are set.

2: Alternate pin assignment for P2B, T3CKI, CCP3 and CCP2 when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are clear.

TABLE 1-2: PIC18(L)F2XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Number		Pin Name	Pin Type	Buffer Type	Description
PDIP, SOIC	QFN				
20	17	VDD	P	—	Positive supply for logic and I/O pins.
8, 19	5, 16	VSS	P	—	Ground reference for logic and I/O pins.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power.

Note 1: Default pin assignment for P2B, T3CKI, CCP3 and CCP2 when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are set.

2: Alternate pin assignment for P2B, T3CKI, CCP3 and CCP2 when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are clear.

TABLE 1-3: PIC18(L)F4XK22 PINOUT I/O DESCRIPTIONS

Pin Number			Pin Name	Pin Type	Buffer Type	Description
PDIP	TQFP	QFN				
2	19	19	RA0/C12IN0-/AN0			
			RA0	I/O	TTL	Digital I/O.
			C12IN0-	I	Analog	Comparators C1 and C2 inverting input.
			AN0	I	Analog	Analog input 0.
3	20	20	RA1/C12IN1-/AN1			
			RA1	I/O	TTL	Digital I/O.
			C12IN1-	I	Analog	Comparators C1 and C2 inverting input.
			AN1	I	Analog	Analog input 1.
4	21	21	RA2/C2IN+/AN2/DACOUT/VREF-			
			RA2	I/O	TTL	Digital I/O.
			C2IN+	I	Analog	Comparator C2 non-inverting input.
			AN2	I	Analog	Analog input 2.
			DACOUT	O	Analog	DAC Reference output.
			VREF-	I	Analog	A/D reference voltage (low) input.
5	22	22	RA3/C1IN+/AN3/VREF+			
			RA3	I/O	TTL	Digital I/O.
			C1IN+	I	Analog	Comparator C1 non-inverting input.
			AN3	I	Analog	Analog input 3.
			VREF+	I	Analog	A/D reference voltage (high) input.
6	23	23	RA4/C1OUT/SRQ/T0CKI			
			RA4	I/O	TTL	Digital I/O.
			C1OUT	O	CMOS	Comparator C1 output.
			SRQ	O	TTL	SR Latch Q output.
			T0CKI	I	ST	Timer0 external clock input.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power.

Note 1: Default pin assignment for P2B, T3CKI, CCP3/P3A and CCP2/P2A when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are set.

2: Alternate pin assignment for P2B, T3CKI, CCP3/P3A and CCP2/P2A when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are clear.

TABLE 1-3: PIC18(L)F4XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Number			Pin Name	Pin Type	Buffer Type	Description
PDIP	TQFP	QFN				
7	24	24	RA5/C2OUT/SRNQ/$\overline{SS1}$/HLVDIN/AN4			
			RA5	I/O	TTL	Digital I/O.
			C2OUT	O	CMOS	Comparator C2 output.
			SRNQ	O	TTL	SR Latch $\overline{Q}$ output.
			$\overline{SS1}$	I	TTL	SPI slave select input (MSSP1).
			HLVDIN	I	Analog	High/Low-Voltage Detect input.
			AN4	I	Analog	Analog input 4.
14	31	33	RA6/CLKO/OSC2			
			RA6	I/O	TTL	Digital I/O.
			CLKO	O	—	In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
			OSC2	O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
13	30	32	RA7/CLKI/OSC1			
			RA7	I/O	TTL	Digital I/O.
			CLKI	I	CMOS	External clock source input. Always associated with pin function OSC1.
			OSC1	I	ST	Oscillator crystal input or external clock source input ST buffer when configured in RC mode; CMOS otherwise.
33	8	9	RB0/INT0/FLT0/SRI/AN12			
			RB0	I/O	TTL	Digital I/O.
			INT0	I	ST	External interrupt 0.
			FLT0	I	ST	PWM Fault input for ECCP Auto-Shutdown.
			SRI	I	ST	SR Latch input.
			AN12	I	Analog	Analog input 12.
34	9	10	RB1/INT1/C12IN3-/AN10			
			RB1	I/O	TTL	Digital I/O.
			INT1	I	ST	External interrupt 1.
			C12IN3-	I	Analog	Comparators C1 and C2 inverting input.
			AN10	I	Analog	Analog input 10.
35	10	11	RB2/INT2/CTED1/AN8			
			RB2	I/O	TTL	Digital I/O.
			INT2	I	ST	External interrupt 2.
			CTED1	I	ST	CTMU Edge 1 input.
			AN8	I	Analog	Analog input 8.
36	11	12	RB3/CTED2/P2A/CCP2/C12IN2-/AN9			
			RB3	I/O	TTL	Digital I/O.
			CTED2	I	ST	CTMU Edge 2 input.
			P2A(2)	O	CMOS	Enhanced CCP2 PWM output.
			CCP2(2)	I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
			C12IN2-	I	Analog	Comparators C1 and C2 inverting input.
			AN9	I	Analog	Analog input 9.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power.

Note 1: Default pin assignment for P2B, T3CKI, CCP3/P3A and CCP2/P2A when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are set.

2: Alternate pin assignment for P2B, T3CKI, CCP3/P3A and CCP2/P2A when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are clear.

TABLE 1-3: PIC18(L)F4XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Number			Pin Name	Pin Type	Buffer Type	Description
PDIP	TQFP	QFN				
37	14	14	RB4/IOC0/T5G/AN11			
			RB4	I/O	TTL	Digital I/O.
			IOC0	I	TTL	Interrupt-on-change pin.
			T5G	I	ST	Timer5 external clock gate input.
			AN11	I	Analog	Analog input 11.
38	15	15	RB5/IOC1/P3A/CCP3/T3CKI/T1G/AN13			
			RB5	I/O	TTL	Digital I/O.
			IOC1	I	TTL	Interrupt-on-change pin.
			P3A(1)	O	CMOS	Enhanced CCP3 PWM output.
			CCP3(1)	I/O	ST	Capture 3 input/Compare 3 output/PWM 3 output.
			T3CKI(2)	I	ST	Timer3 clock input.
			T1G	I	ST	Timer1 external clock gate input.
			AN13	I	Analog	Analog input 13.
39	16	16	RB6/IOC2/PGC			
			RB6	I/O	TTL	Digital I/O.
			IOC2	I	TTL	Interrupt-on-change pin.
			PGC	I/O	ST	In-Circuit Debugger and ICSP™ programming clock pin.
40	17	17	RB7/IOC3/PGD			
			RB7	I/O	TTL	Digital I/O.
			IOC3	I	TTL	Interrupt-on-change pin.
			PGD	I/O	ST	In-Circuit Debugger and ICSP™ programming data pin.
15	32	34	RC0/P2B/T3CKI/T3G/T1CKI/SOSCO			
			RC0	I/O	TTL	Digital I/O.
			P2B(2)	O	CMOS	Enhanced CCP1 PWM output.
			T3CKI(1)	I	ST	Timer3 clock input.
			T3G	I	ST	Timer3 external clock gate input.
			T1CKI	I	ST	Timer1 clock input.
			SOSCO	O	—	Secondary oscillator output.
16	35	35	RC1/P2A/CCP2/SOSCI			
			RC1	I/O	TTL	Digital I/O.
			P2A(1)	O	CMOS	Enhanced CCP2 PWM output.
			CCP2(1)	I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
			SOSCI	I	Analog	Secondary oscillator input.
17	36	36	RC2/CTPLS/P1A/CCP1/T5CKI/AN14			
			RC2	I/O	TTL	Digital I/O.
			CTPLS	O	—	CTMU pulse generator output.
			P1A	O	CMOS	Enhanced CCP1 PWM output.
			CCP1	I/O	ST	Capture 1 input/Compare 1 output/PWM 1 output.
			T5CKI	I	ST	Timer5 clock input.
			AN14	I	Analog	Analog input 14.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power.

Note 1: Default pin assignment for P2B, T3CKI, CCP3/P3A and CCP2/P2A when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are set.

2: Alternate pin assignment for P2B, T3CKI, CCP3/P3A and CCP2/P2A when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are clear.

TABLE 1-3: PIC18(L)F4XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Number			Pin Name	Pin Type	Buffer Type	Description
PDIP	TQFP	QFN				
18	37	37	RC3/SCK1/SCL1/AN15			
			RC3	I/O	TTL	Digital I/O.
			SCK1	I/O	ST	Synchronous serial clock input/output for SPI mode (MSSP2).
			SCL1	I/O	ST	Synchronous serial clock input/output for I^2C™ mode (MSSP2).
			AN15	I	Analog	Analog input 15.
23	42	42	RC4/SDI1/SDA1/AN16			
			RC4	I/O	TTL	Digital I/O.
			SDI1	I	ST	SPI data in (MSSP1).
			SDA1	I/O	ST	I^2C™ data I/O (MSSP1).
			AN16	I	Analog	Analog input 16.
24	43	43	RC5/SDO1/AN17			
			RC5	I/O	TTL	Digital I/O.
			SDO1	O	—	SPI data out (MSSP1).
			AN17	I	Analog	Analog input 17.
25	44	44	RC6/TX1/CK1/AN18			
			RC6	I/O	TTL	Digital I/O.
			TX1	O	—	EUSART 1 asynchronous transmit.
			CK1	I/O	ST	EUSART 1 synchronous clock (see related RXx/DTx).
			AN18	I	Analog	Analog input 18.
26	1	1	RC7/RX1/DT1/AN19			
			RC7	I/O	TTL	Digital I/O.
			RX1	I	ST	EUSART 1 asynchronous receive.
			DT1	I/O	ST	EUSART 1 synchronous data (see related TXx/CKx).
			AN19	I	Analog	Analog input 19.
19	38	38	RD0/SCK2/SCL2/AN20			
			RD0	I/O	TTL	Digital I/O.
			SCK2	I/O	ST	Synchronous serial clock input/output for SPI mode (MSSP2).
			SCL2	I/O	ST	Synchronous serial clock input/output for I^2C™ mode (MSSP2).
			AN20	I	Analog	Analog input 20.
20	39	39	RD1/CCP4/SDI2/SDA2/AN21			
			RD1	I/O	TTL	Digital I/O.
			CCP4	I/O	ST	Capture 4 input/Compare 4 output/PWM 4 output.
			SDI2	I	ST	SPI data in (MSSP2).
			SDA2	I/O	ST	I^2C™ data I/O (MSSP2).
			AN21	I	Analog	Analog input 21.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power.

Note 1: Default pin assignment for P2B, T3CKI, CCP3/P3A and CCP2/P2A when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are set.

2: Alternate pin assignment for P2B, T3CKI, CCP3/P3A and CCP2/P2A when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are clear.

TABLE 1-3: PIC18(L)F4XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Number			Pin Name	Pin Type	Buffer Type	Description
PDIP	TQFP	QFN				
21	40	40	RD2/P2B/AN22			
			RD2	I/O	TTL	Digital I/O
			P2B(1)	O	CMOS	Enhanced CCP2 PWM output.
			AN22	I	Analog	Analog input 22.
22	41	41	RD3/P2C/$\overline{SS2}$/AN23			
			RD3	I/O	TTL	Digital I/O.
			P2C	O	CMOS	Enhanced CCP2 PWM output.
			$\overline{SS2}$	I	TTL	SPI slave select input (MSSP2).
			AN23	I	Analog	Analog input 23.
27	2	2	RD4/P2D/SDO2/AN24			
			RD4	I/O	TTL	Digital I/O.
			P2D	O	CMOS	Enhanced CCP2 PWM output.
			SDO2	O	—	SPI data out (MSSP2).
			AN24	I	Analog	Analog input 24.
28	3	3	RD5/P1B/AN25			
			RD5	I/O	TTL	Digital I/O.
			P1B	O	CMOS	Enhanced CCP1 PWM output.
			AN25	I	Analog	Analog input 25.
29	4	4	RD6/P1C/TX2/CK2/AN26			
			RD6	I/O	TTL	Digital I/O.
			P1C	O	CMOS	Enhanced CCP1 PWM output.
			TX2	O	—	EUSART 2 asynchronous transmit.
			CK2	I/O	ST	EUSART 2 synchronous clock (see related RXx/DTx).
			AN26	I	Analog	Analog input 26.
30	5	5	RD7/P1D/RX2/DT2/AN27			
			RD7	I/O	TTL	Digital I/O.
			P1D	O	CMOS	Enhanced CCP1 PWM output.
			RX2	I	ST	EUSART 2 asynchronous receive.
			DT2	I/O	ST	EUSART 2 synchronous data (see related TXx/CKx).
			AN27	I	Analog	Analog input 27.
8	25	25	RE0/P3A/CCP3/AN5			
			RE0	I/O	TTL	Digital I/O.
			P3A(2)	O	CMOS	Enhanced CCP3 PWM output.
			CCP3(2)	I/O	ST	Capture 3 input/Compare 3 output/PWM 3 output.
			AN5	I	Analog	Analog input 5.
9	26	26	RE1/P3B/AN6			
			RE1	I/O	TTL	Digital I/O.
			P3B	O	CMOS	Enhanced CCP3 PWM output.
			AN6	I	Analog	Analog input 6.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power.

Note 1: Default pin assignment for P2B, T3CKI, CCP3/P3A and CCP2/P2A when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are set.

2: Alternate pin assignment for P2B, T3CKI, CCP3/P3A and CCP2/P2A when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are clear.

TABLE 1-3: PIC18(L)F4XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Number			Pin Name	Pin Type	Buffer Type	Description
PDIP	TQFP	QFN				
10	27	27	RE2/CCP5/AN7			
			RE2	I/O	TTL	Digital I/O.
			CCP5	I/O	ST	Capture 5 input/Compare 5 output/PWM 5 output
			AN7	I	Analog	Analog input 7.
1	18	18	RE3/VPP/$\overline{\text{MCLR}}$			
			RE3	I	ST	Digital input.
			VPP	P		Programming voltage input.
			$\overline{\text{MCLR}}$	I	ST	Active-low Master Clear (device Reset) input.
11,32	7,28	7,8, 28,29	VDD	P	—	Positive supply for logic and I/O pins.
12,31	6,29	6,30,31	VSS	P	—	Ground reference for logic and I/O pins.
	12,13, 33,34	13	NC			

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power.

Note 1: Default pin assignment for P2B, T3CKI, CCP3/P3A and CCP2/P2A when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are set.

2: Alternate pin assignment for P2B, T3CKI, CCP3/P3A and CCP2/P2A when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are clear.

TABLE 5-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F2X/4XK22 DEVICES

Address	Name	Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FD7h	TMR0H	FAFh	SPBRG1	F87h	—(2)	F5Fh	CCPR3H
FFEh	TOSH	FD6h	TMR0L	FAEh	RCREG1	F86h	—(2)	F5Eh	CCPR3L
FFDh	TOSL	FD5h	T0CON	FADh	TXREG1	F85h	—(2)	F5Dh	CCP3CON
FFCh	STKPTR	FD4h	—(2)	FACh	TXSTA1	F84h	PORTE	F5Ch	PWM3CON
FFBh	PCLATU	FD3h	OSCCON	FABh	RCSTA1	F83h	PORTD(3)	F5Bh	ECCP3AS
FFAh	PCLATH	FD2h	OSCCON2	FAAh	EEADRH(4)	F82h	PORTC	F5Ah	PSTR3CON
FF9h	PCL	FD1h	WDTCON	FA9h	EEADR	F81h	PORTB	F59h	CCPR4H
FF8h	TBLPTRU	FD0h	RCON	FA8h	EEDATA	F80h	PORTA	F58h	CCPR4L
FF7h	TBLPTRH	FCFh	TMR1H	FA7h	EECON2(1)	F7Fh	IPR5	F57h	CCP4CON
FF6h	TBLPTRL	FCEh	TMR1L	FA6h	EECON1	F7Eh	PIR5	F56h	CCPR5H
FF5h	TABLAT	FCDh	T1CON	FA5h	IPR3	F7Dh	PIE5	F55h	CCPR5L
FF4h	PRODH	FCCh	T1GCON	FA4h	PIR3	F7Ch	IPR4	F54h	CCP5CON
FF3h	PRODL	FCBh	SSP1CON3	FA3h	PIE3	F7Bh	PIR4	F53h	TMR4
FF2h	INTCON	FCAh	SSP1MSK	FA2h	IPR2	F7Ah	PIE4	F52h	PR4
FF1h	INTCON2	FC9h	SSP1BUF	FA1h	PIR2	F79h	CM1CON0	F51h	T4CON
FF0h	INTCON3	FC8h	SSP1ADD	FA0h	PIE2	F78h	CM2CON0	F50h	TMR5H
FEFh	INDF0(1)	FC7h	SSP1STAT	F9Fh	IPR1	F77h	CM2CON1	F4Fh	TMR5L
FEEh	POSTINC0(1)	FC6h	SSP1CON1	F9Eh	PIR1	F76h	SPBRGH2	F4Eh	T5CON
FEDh	POSTDEC0(1)	FC5h	SSP1CON2	F9Dh	PIE1	F75h	SPBRG2	F4Dh	T5GCON
FECh	PREINC0(1)	FC4h	ADRESH	F9Ch	HLVDCON	F74h	RCREG2	F4Ch	TMR6
FEBh	PLUSW0(1)	FC3h	ADRESL	F9Bh	OSCTUNE	F73h	TXREG2	F4Bh	PR6
FEAh	FSR0H	FC2h	ADCON0	F9Ah	—(2)	F72h	TXSTA2	F4Ah	T6CON
FE9h	FSR0L	FC1h	ADCON1	F99h	—(2)	F71h	RCSTA2	F49h	CCPTMRS0
FE8h	WREG	FC0h	ADCON2	F98h	—(2)	F70h	BAUDCON2	F48h	CCPTMRS1
FE7h	INDF1(1)	FBFh	CCPR1H	F97h	—(2)	F6Fh	SSP2BUF	F47h	SRCON0
FE6h	POSTINC1(1)	FBEh	CCPR1L	F96h	TRISE	F6Eh	SSP2ADD	F46h	SRCON1
FE5h	POSTDEC1(1)	FBDh	CCP1CON	F95h	TRISD(3)	F6Dh	SSP2STAT	F45h	CTMUCONH
FE4h	PREINC1(1)	FBCh	TMR2	F94h	TRISC	F6Ch	SSP2CON1	F44h	CTMUCONL
FE3h	PLUSW1(1)	FBBh	PR2	F93h	TRISB	F6Bh	SSP2CON2	F43h	CTMUICON
FE2h	FSR1H	FBAh	T2CON	F92h	TRISA	F6Ah	SSP2MSK	F42h	VREFCON0
FE1h	FSR1L	FB9h	PSTR1CON	F91h	—(2)	F69h	SSP2CON3	F41h	VREFCON1
FE0h	BSR	FB8h	BAUDCON1	F90h	—(2)	F68h	CCPR2H	F40h	VREFCON2
FDFh	INDF2(1)	FB7h	PWM1CON	F8Fh	—(2)	F67h	CCPR2L	F3Fh	PMD0
FDEh	POSTINC2(1)	FB6h	ECCP1AS	F8Eh	—(2)	F66h	CCP2CON	F3Eh	PMD1
FDDh	POSTDEC2(1)	FB5h	—(2)	F8Dh	LATE(3)	F65h	PWM2CON	F3Dh	PMD2
FDCh	PREINC2(1)	FB4h	T3GCON	F8Ch	LATD(3)	F64h	ECCP2AS	F3Ch	ANSELE
FDBh	PLUSW2(1)	FB3h	TMR3H	F8Bh	LATC	F63h	PSTR2CON	F3Bh	ANSELD
FDAh	FSR2H	FB2h	TMR3L	F8Ah	LATB	F62h	IOCB	F3Ah	ANSELC
FD9h	FSR2L	FB1h	T3CON	F89h	LATA	F61h	WPUB	F39h	ANSELB
FD8h	STATUS	FB0h	SPBRGH1	F88h	—(2)	F60h	SLRCON	F38h	ANSELA

Note 1: This is not a physical register.
2: Unimplemented registers are read as '0'.
3: PIC18(L)F4XK22 devices only.
4: PIC18(L)F26K22 and PIC18(L)F46K22 devices only.

TABLE 5-2: REGISTER FILE SUMMARY FOR PIC18(L)F2X/4XK22 DEVICES

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
FFFh	TOSU	—	—	—	Top-of-Stack, Upper Byte (TOS<20:16>)					---0 0000
FFEh	TOSH	Top-of-Stack, High Byte (TOS<15:8>)								0000 0000
FFDh	TOSL	Top-of-Stack, Low Byte (TOS<7:0>)								0000 0000
FFCh	STKPTR	STKFUL	STKUNF	—	STKPTR<4:0>					00-0 0000
FFBh	PCLATU	—	—	—	Holding Register for PC<20:16>					---0 0000
FFAh	PCLATH	Holding Register for PC<15:8>								0000 0000
FF9h	PCL	Holding Register for PC<7:0>								0000 0000
FF8h	TBLPTRU	—	—	Program Memory Table Pointer Upper Byte(TBLPTR<21:16>)						--00 0000
FF7h	TBLPTRH	Program Memory Table Pointer High Byte(TBLPTR<15:8>)								0000 0000
FF6h	TBLPTRL	Program Memory Table Pointer Low Byte(TBLPTR<7:0>)								0000 0000
FF5h	TABLAT	Program Memory Table Latch								0000 0000
FF4h	PRODH	Product Register, High Byte								xxxx xxxx
FF3h	PRODL	Product Register, Low Byte								xxxx xxxx
FF2h	INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x
FF1h	INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	1111 -1-1
FF0h	INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	11-0 0-00
FEFh	INDF0	Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register)								---- ----
FEEh	POSTINC0	Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register)								---- ----
FEDh	POSTDEC0	Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register)								---- ----
FECh	PREINC0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register)								---- ----
FEBh	PLUSW0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) – value of FSR0 offset by W								---- ----
FEAh	FSR0H	—	—	—	—	Indirect Data Memory Address Pointer 0, High Byte				---- 0000
FE9h	FSR0L	Indirect Data Memory Address Pointer 0, Low Byte								xxxx xxxx
FE8h	WREG	Working Register								xxxx xxxx
FE7h	INDF1	Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register)								---- ----
FE6h	POSTINC1	Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register)								---- ----
FE5h	POSTDEC1	Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register)								---- ----
FE4h	PREINC1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register)								---- ----
FE3h	PLUSW1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) – value of FSR1 offset by W								---- ----
FE2h	FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1, High Byte				---- 0000
FE1h	FSR1L	Indirect Data Memory Address Pointer 1, Low Byte								xxxx xxxx
FE0h	BSR	—	—	—	—	Bank Select Register				---- 0000
FDFh	INDF2	Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register)								---- ----
FDEh	POSTINC2	Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register)								---- ----
FDDh	POSTDEC2	Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register)								---- ----
FDCh	PREINC2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register)								---- ----
FDBh	PLUSW2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – value of FSR2 offset by W								---- ----
FDAh	FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2, High Byte				---- 0000
FD9h	FSR2L	Indirect Data Memory Address Pointer 2, Low Byte								xxxx xxxx
FD8h	STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx
FD7h	TMR0H	Timer0 Register, High Byte								0000 0000
FD6h	TMR0L	Timer0 Register, Low Byte								xxxx xxxx
FD5h	T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS<2:0>			1111 1111
FD3h	OSCCON	IDLEN	IRCF<2:0>			OSTS	HFIOFS	SCS<1:0>		0011 q000
FD2h	OSCCON2	PLLRDY	SOSCRUN	—	MFIOSEL	SOSCGO	PRISD	MFIOFS	LFIOFS	00-0 01x0

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: PIC18(L)F4XK22 devices only.

2: PIC18(L)F2XK22 devices only.

3: PIC18(L)F23/24K22 and PIC18(L)F43/44K22 devices only.

4: PIC18(L)F26K22 and PIC18(L)F46K22 devices only.

TABLE 5-2: REGISTER FILE SUMMARY FOR PIC18(L)F2X/4XK22 DEVICES

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
FD1h	WDTCON	—	—	—	—	—	—	—	SWDTEN	---- ---0
FD0h	RCON	IPEN	SBOREN	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	01-1 1100
FCFh	TMR1H	Timer1 Register, High Byte								xxxx xxxx
FCEh	TMR1L	Timer1 Register, Low Byte								xxxx xxxx
FCDh	T1CON	TMR1CS<1:0>		T1CKPS<1:0>		T1SOSCEN	$\overline{T1SYNC}$	T1RD16	TMR1ON	0000 0000
FCCh	T1GCON	TMR1GE	T1GPOL	T1GTM	T1GSPM	T1GGO/$\overline{DONE}$	T1GVAL	T1GSS<1:0>		0000 xx00
FCBh	SSP1CON3	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN	0000 0000
FCAh	SSP1MSK	SSP1 MASK Register bits								1111 1111
FC9h	SSP1BUF	SSP1 Receive Buffer/Transmit Register								xxxx xxxx
FC8h	SSP1ADD	SSP1 Address Register in I^2C Slave Mode. SSP1 Baud Rate Reload Register in I^2C Master Mode								0000 0000
FC7h	SSP1STAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000
FC6h	SSP1CON1	WCOL	SSPOV	SSPEN	CKP	SSPM<3:0>				0000 0000
FC5h	SSP1CON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000
FC4h	ADRESH	A/D Result, High Byte								xxxx xxxx
FC3h	ADRESL	A/D Result, Low Byte								xxxx xxxx
FC2h	ADCON0	—	CHS<4:0>					GO/$\overline{DONE}$	ADON	--00 0000
FC1h	ADCON1	TRIGSEL	—	—	—	PVCFG<1:0>		NVCFG<1:0>		0--- 0000
FC0h	ADCON2	ADFM	—	ACQT<2:0>		ADCS<2:0>				0-00 0000
FBFh	CCPR1H	Capture/Compare/PWM Register 1, High Byte								xxxx xxxx
FBEh	CCPR1L	Capture/Compare/PWM Register 1, Low Byte								xxxx xxxx
FBDh	CCP1CON	P1M<1:0>		DC1B<1:0>		CCP1M<3:0>				0000 0000
FBCh	TMR2	Timer2 Register								0000 0000
FBBh	PR2	Timer2 Period Register								1111 1111
FBAh	T2CON	—	T2OUTPS<3:0>				TMR2ON	T2CKPS<1:0>		-000 0000
FB9h	PSTR1CON	—	—	—	STR1SYNC	STR1D	STR1C	STR1B	STR1A	---0 0001
FB8h	BAUDCON1	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	0100 0-00
FB7h	PWM1CON	P1RSEN	P1DC<6:0>							0000 0000
FB6h	ECCP1AS	CCP1ASE	CCP1AS<2:0>			P1SSAC<1:0>		P1SSBD<1:0>		0000 0000
FB4h	T3GCON	TMR3GE	T3GPOL	T3GTM	T3GSPM	T3GGO/$\overline{DONE}$	T3GVAL	T3GSS		0000 0x00
FB3h	TMR3H	Timer3 Register, High Byte								xxxx xxxx
FB2h	TMR3L	Timer3 Register, Low Byte								xxxx xxxx
FB1h	T3CON	TMR3CS<1:0>		T3CKPS<1:0>		T3SOSCEN	$\overline{T3SYNC}$	T3RD16	TMR3ON	0000 0000
FB0h	SPBRGH1	EUSART1 Baud Rate Generator, High Byte								0000 0000
FAFh	SPBRG1	EUSART1 Baud Rate Generator, Low Byte								0000 0000
FAEh	RCREG1	EUSART1 Receive Register								0000 0000
FADh	TXREG1	EUSART1 Transmit Register								0000 0000
FACh	TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010
FABh	RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x
FAAh	EEADRH(5)	—	—	—	—	—	—	EEADR<9:8>		---- --00
FA9h	EEADR	EEADR<7:0>								0000 0000
FA8h	EEDATA	EEPROM Data Register								0000 0000
FA7h	EECON2	EEPROM Control Register 2 (not a physical register)								---- --00
FA6h	EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	xx-0 x000
FA5h	IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	CTMUIP	TMR5GIP	TMR3GIP	TMR1GIP	0000 0000
FA4h	PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	CTMUIF	TMR5GIF	TMR3GIF	TMR1GIF	0000 0000
FA3h	PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	CTMUIE	TMR5GIE	TMR3GIE	TMR1GIE	0000 0000

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: PIC18(L)F4XK22 devices only.
2: PIC18(L)F2XK22 devices only.
3: PIC18(L)F23/24K22 and PIC18(L)F43/44K22 devices only.
4: PIC18(L)F26K22 and PIC18(L)F46K22 devices only.

TABLE 5-2: REGISTER FILE SUMMARY FOR PIC18(L)F2X/4XK22 DEVICES

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
FA2h	IPR2	OSCFIP	C1IP	C2IP	EEIP	BCL1IP	HLVDIP	TMR3IP	CCP2IP	1111 1111
FA1h	PIR2	OSCFIF	C1IF	C2IF	EEIF	BCL1IF	HLVDIF	TMR3IF	CCP2IF	0000 0000
FA0h	PIE2	OSCFIE	C1IE	C2IE	EEIE	BCL1IE	HLVDIE	TMR3IE	CCP2IE	0000 0000
F9Fh	IPR1	—	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP	-111 1111
F9Eh	PIR1	—	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	-000 0000
F9Dh	PIE1	—	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	-000 0000
F9Ch	HLVDCON	VDIRMAG	BGVST	IRVST	HLVDEN	HLVDL<3:0>				0000 0000
F9Bh	OSCTUNE	INTSRC	PLLEN	TUN<5:0>						00xx xxxx
F96h	TRISE	WPUE3	—	—	—	—	TRISE2(1)	TRISE1(1)	TRISE0(1)	1--- -111
F95h	TRISD(1)	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0	1111 1111
F94h	TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	1111 1111
F93h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111
F92h	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1111 1111
F8Dh	LATE(1)	—	—	—	—	—	LATE2	LATE1	LATE0	---- -xxx
F8Ch	LATD(1)	LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0	xxxx xxxx
F8Bh	LATC	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0	xxxx xxxx
F8Ah	LATB	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	xxxx xxxx
F89h	LATA	LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0	xxxx xxxx
F84h	PORTE(2)	—	—	—	—	RE3	—	—	—	---- x---
	PORTE(1)	—	—	—	—	RE3	RE2	RE1	RE0	---- x000
F83h	PORTD(1)	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	0000 0000
F82h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	0000 00xx
F81h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxx0 0000
F80h	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xx0x 0000
F7Fh	IPR5	—	—	—	—	—	TMR6IP	TMR5IP	TMR4IP	---- -111
F7Eh	PIR5	—	—	—	—	—	TMR6IF	TMR5IF	TMR4IF	---- -111
F7Dh	PIE5	—	—	—	—	—	TMR6IE	TMR5IE	TMR4IE	---- -000
F7Ch	IPR4	—	—	—	—	—	CCP5IP	CCP4IP	CCP3IP	---- -000
F7Bh	PIR4	—	—	—	—	—	CCP5IF	CCP4IF	CCP3IF	---- -000
F7Ah	PIE4	—	—	—	—	—	CCP5IE	CCP4IE	CCP3IE	---- -000
F79h	CM1CON0	C1ON	C1OUT	C1OE	C1POL	C1SP	C1R	C1CH<1:0>		0000 1000
F78h	CM2CON0	C2ON	C2OUT	C2OE	C2POL	C2SP	C2R	C2CH<1:0>		0000 1000
F77h	CM2CON1	MC1OUT	MC2OUT	C1RSEL	C2RSEL	C1HYS	C2HYS	C1SYNC	C2SYNC	0000 0000
F76h	SPBRGH2	EUSART2 Baud Rate Generator, High Byte								0000 0000
F75h	SPBRG2	EUSART2 Baud Rate Generator, Low Byte								0000 0000
F74h	RCREG2	EUSART2 Receive Register								0000 0000
F73h	TXREG2	EUSART2 Transmit Register								0000 0000
F72h	TXSTA2	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010
F71h	RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x
F70h	BAUDCON2	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	01x0 0-00
F6Fh	SSP2BUF	SSP2 Receive Buffer/Transmit Register								xxxx xxxx
F6Eh	SSP2ADD	SSP2 Address Register in I2C Slave Mode. SSP2 Baud Rate Reload Register in I2C Master Mode								0000 0000
F6Dh	SSP2STAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000
F6Ch	SSP2CON1	WCOL	SSPOV	SSPEN	CKP	SSPM<3:0>				0000 0000
F6Bh	SSP2CON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000
F6Ah	SSP2MSK	SSP1 MASK Register bits								1111 1111
F69h	SSP2CON3	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN	0000 0000

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: PIC18(L)F4XK22 devices only.
2: PIC18(L)F2XK22 devices only.
3: PIC18(L)F23/24K22 and PIC18(L)F43/44K22 devices only.
4: PIC18(L)F26K22 and PIC18(L)F46K22 devices only.

TABLE 5-2: REGISTER FILE SUMMARY FOR PIC18(L)F2X/4XK22 DEVICES

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
F68h	CCPR2H	Capture/Compare/PWM Register 2, High Byte								xxxx xxxx
F67h	CCPR2L	Capture/Compare/PWM Register 2, Low Byte								xxxx xxxx
F66h	CCP2CON	P2M<1:0>		DC2B<1:0>		CCP2M<3:0>				0000 0000
F65h	PWM2CON	P2RSEN	P2DC<6:0>							0000 0000
F64h	ECCP2AS	CCP2ASE	CCP2AS<2:0>			P2SSAC<1:0>		P2SSBD<1:0>		0000 0000
F63h	PSTR2CON	—	—	—	STR2SYNC	STR2D	STR2C	STR2B	STR2A	---0 0001
F62h	IOCB	IOCB7	IOCB6	IOCB5	IOCB4	—	—	—	—	1111 ----
F61h	WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	1111 1111
F60h	SLRCON(2)	—	—	—	—	—	SLRC	SLRB	SLRA	---- -111
	SLRCON(1)	—	—	—	SLRE	SLRD	SLRC	SLRB	SLRA	---1 1111
F5Fh	CCPR3H	Capture/Compare/PWM Register 3, High Byte								xxxx xxxx
F5Eh	CCPR3L	Capture/Compare/PWM Register 3, Low Byte								xxxx xxxx
F5Dh	CCP3CON	P3M<1:0>		DC3B<1:0>		CCP3M<3:0>				0000 0000
F5Ch	PWM3CON	P3RSEN	P3DC<6:0>							0000 0000
F5Bh	ECCP3AS	CCP3ASE	CCP3AS<2:0>			P3SSAC<1:0>		P3SSBD<1:0>		0000 0000
F5Ah	PSTR3CON	—	—	—	STR3SYNC	STR3D	STR3C	STR3B	STR3A	---0 0001
F59h	CCPR4H	Capture/Compare/PWM Register 4, High Byte								xxxx xxxx
F58h	CCPR4L	Capture/Compare/PWM Register 4, Low Byte								xxxx xxxx
F57h	CCP4CON	—	—	DC4B<1:0>		CCP4M<3:0>				--00 0000
F56h	CCPR5H	Capture/Compare/PWM Register 5, High Byte								xxxx xxxx
F55h	CCPR5L	Capture/Compare/PWM Register 5, Low Byte								xxxx xxxx
F54h	CCP5CON	—	—	DC5B<1:0>		CCP5M<3:0>				--00 0000
F53h	TMR4	Timer4 Register								0000 0000
F52h	PR4	Timer4 Period Register								1111 1111
F51h	T4CON	—	T4OUTPS<3:0>				TMR4ON	T4CKPS<1:0>		-000 0000
F50h	TMR5H	Timer5 Register, High Byte								0000 0000
F4Fh	TMR5L	Timer5 Register, Low Byte								0000 0000
F4Eh	T5CON	TMR5CS<1:0>		T5CKPS<1:0>		T5SOSCEN	T5SYNC	T5RD16	TMR5ON	0000 0000
F4Dh	T5GCON	TMR5GE	T5GPOL	T5GTM	T5GSPM	T5GGO/DONE	T5GVAL	T5GSS		0000 0x00
F4Ch	TMR6	Timer6 Register								0000 0000
F4Bh	PR6	Timer6 Period Register								1111 1111
F4Ah	T6CON	—	T6OUTPS<3:0>				TMR6ON	T6CKPS<1:0>		-000 0000
F49h	CCPTMRS0	C3TSEL<1:0>		—	C2TSEL<1:0>		—	C1TSEL<1:0>		00-0 0-00
F48h	CCPTMRS1	—	—	—	—	C5TSEL<1:0>		C4TSEL<1:0>		---- 0000
F47h	SRCON0	SRLEN	SRCLK<2:0>			SRQEN	SRNQEN	SRPS	SRPR	0000 0000
F46h	SRCON1	SRSPE	SRSCKE	SRSC2E	SRSC1E	SRRPE	SRRCKE	SRRC2E	SRRC1E	0000 0000
F45h	CTMUCONH	CTMUEN	—	CTMUSIDL	TGEN	EDGEN	EDGSEQEN	IDISSEN	CTTRIG	0000 0000
F44h	CTMUCONL	EDG2POL	EDG2SEL<1:0>		EDG1POL	EDG1SEL<1:0>		EDG2STAT	EDG1STAT	0000 0000
F43h	CTMUICON	ITRIM<5:0>						IRNG<1:0>		0000 0000
F42h	VREFCON0	FVREN	FVRST	FVRS<1:0>		—	—	—	—	0001 ----
F41h	VREFCON1	DACEN	DACLPS	DACOE	—	DACPSS<1:0>		—	DACNSS	000- 00-0
F40h	VREFCON2	—	—	—	DACR<4:0>					---0 0000
F3Fh	PMD0	UART2MD	UART1MD	TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	0000 0000
F3Eh	PMD1	MSSP2MD	MSSP1MD	—	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	00-0 0000
F3Dh	PMD2	—	—	—	—	CTMUMD	CMP2MD	CMP1MD	ADCMD	---- 0000
F3Ch	ANSELE(1)	—	—	—	—	—	ANSE2	ANSE1	ANSE0	---- -111
F3Bh	ANSELD(1)	ANSD7	ANSD6	ANSD5	ANSD4	ANSD3	ANSD2	ANSD1	ANSD0	1111 1111

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: PIC18(L)F4XK22 devices only.
2: PIC18(L)F2XK22 devices only.
3: PIC18(L)F23/24K22 and PIC18(L)F43/44K22 devices only.
4: PIC18(L)F26K22 and PIC18(L)F46K22 devices only.

TABLE 5-2: REGISTER FILE SUMMARY FOR PIC18(L)F2X/4XK22 DEVICES

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
F3Ah	ANSELC	ANSC7	ANSC6	ANSC5	ANSC4	ANSC3	ANSC2	—	—	1111 11--
F39h	ANSELB	—	—	ANSB5	ANSB4	ANSB3	ANSB2	ANSB1	ANSB0	--11 1111
F38h	ANSELA	—	—	ANSA5	—	ANSA3	ANSA2	ANSA1	ANSA0	--1- 1111

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: PIC18(L)F4XK22 devices only.

2: PIC18(L)F2XK22 devices only.

3: PIC18(L)F23/24K22 and PIC18(L)F43/44K22 devices only.

4: PIC18(L)F26K22 and PIC18(L)F46K22 devices only.

