

Universidad de Costa Rica

Facultad de Ingeniería

Escuela de Ingeniería Eléctrica

IE0624 – Laboratorio de Microcontroladores

I ciclo 2025

Laboratorio 1

Gabriel Gamboa Vargas Carné B73098

Grupo 001

Profesor: MSc. Marco Villalta Fallas

25 de marzo

1. Introducción

1.1. Resumen

El microcontrolador PIC12F683, fabricado por Microchip Technology, pertenece a la familia de microcontroladores PIC y opera con una arquitectura de 8 bits. Su tamaño reducido y eficiencia energética lo convierten en una excelente opción para sistemas de control en el ámbito electrónico.

En este proyecto se emplearon los pines de salida del PIC12F683 para controlar un conjunto de LEDs que simulan las diferentes caras de un dado. Cada LED representó un número del 1 al 6, mientras que el microcontrolador generó números aleatorios para imitar el lanzamiento de un dado. Al llevar a cabo la simulación, los LEDs se encendieron conforme al valor generado, logrando reproducir visualmente el comportamiento de un dado real.

Código fuente de este proyecto <https://github.com/GaboUCR/Microcontroladores-Labs.git>.

2. Nota teórica

Un microcontrolador es un dispositivo altamente integrado que incorpora, en un solo chip, la mayoría o la totalidad de los componentes necesarios para ejecutar funciones de control dentro de una aplicación. En el pasado, estas funciones se realizaban mediante circuitos lógicos y dispositivos voluminosos. Con el tiempo, la aparición de los microprocesadores permitió reducir considerablemente el tamaño del sistema, hasta el punto de que el controlador completo podía montarse en una pequeña placa de circuito impreso.

Con el avance de la miniaturización, se logró integrar todos los elementos necesarios en un único chip. Al centrarse exclusivamente en las funciones específicas de la tarea, los microcontroladores presentan un costo bajo. Estos dispositivos suelen incluir instrucciones para la manipulación de bits, permiten un acceso directo y sencillo a las entradas y salidas (I/O), y ofrecen un manejo eficiente de interrupciones [2].

Los microcontroladores se utilizan comúnmente en una amplia gama de dispositivos, como electrodomésticos (hornos microondas, refrigeradores, televisores, videograbadoras, equipos de audio), computadoras y periféricos (impresoras láser, módems, unidades de disco), automóviles (sistemas de control de motor, diagnóstico, climatización), sistemas de control ambiental (en hogares, fábricas, edificios), así como en aplicaciones de instrumentación, aeroespaciales y muchas otras más.

Para la realización del laboratorio que se presenta a continuación, se emplean diversos componentes que se describen en esta sección.

2.1. PIC12f683

El microcontrolador PIC es un circuito integrado lo suficientemente compacto como para caber en la palma de la mano. A diferencia de los microprocesadores tradicionales, que requieren varios circuitos integrados independientes —como el procesador central (CPU), una memoria de programa tipo EPROM, memoria RAM y una interfaz de entrada/salida—, los microcontroladores PIC integran todas estas funciones en un único encapsulado. Esta integración los convierte en una solución económica, eficiente y de fácil uso.

Los microcontroladores PIC pueden funcionar como el “cerebro” de numerosos productos, gracias a su versatilidad. Para poder manejar distintos dispositivos, es necesario establecer una conexión

adecuada con el microcontrolador. Esta sección está diseñada para facilitar ese proceso, especialmente a quienes tienen conocimientos limitados en electrónica, ayudándolos a llevar a cabo con éxito las tareas de interfaz [4].

2.2. Resistores

El resistor es el componente del circuito encargado de representar la oposición al paso de corriente eléctrica en un material. En la fabricación de circuitos, estos dispositivos suelen estar hechos de aleaciones metálicas o materiales a base de carbono.

En la figura 1 se muestra el símbolo del resistor. La letra R representa su resistencia. Este componente es considerado el elemento pasivo más básico dentro de un circuito eléctrico [3].



Figura 1: Símbolo de circuito de un resistor [3].

2.3. LED

El diodo emisor de luz (LED, por sus siglas en inglés) es un componente electrónico que emite luz visible o infrarroja al ser energizado. Aunque no siempre sea perceptible, los LED que emiten en el espectro infrarrojo tienen múltiples aplicaciones donde la emisión de luz visible no es conveniente. Estas aplicaciones abarcan desde sistemas de seguridad, automatización industrial y acoplamientos ópticos, hasta controles de acceso como los utilizados en portones automáticos o sistemas de entretenimiento doméstico, donde el LED infrarrojo actúa como el emisor del control remoto [5].



Figura 2: Ejemplo de diodo emisor de luz (LED) [5].

Como se aprecia en la Figura 2, el LED puede identificarse tanto por su apariencia física como por el símbolo utilizado en esquemas electrónicos.

2.4. Pulsador

Los pulsadores son un tipo de interruptor que permiten o interrumpen el paso de corriente únicamente mientras están siendo presionados. Existen dos variantes: Normalmente Cerrado (NC) y Normalmente Abierto (NO). En el tipo NO, el circuito permanece abierto y solo se cierra cuando se presiona el botón; en cambio, en el tipo NC, el circuito está cerrado por defecto y se interrumpe momentáneamente al presionar el pulsador [1].

Una característica importante a considerar es el rebote mecánico que se produce al accionar el botón, lo que puede generar múltiples transiciones indeseadas. Para mitigar este efecto, se pueden aplicar soluciones tanto por software, introduciendo un pequeño retardo en el código, como por hardware, añadiendo un capacitor que actúe como filtro, cuya capacitancia dependerá de la resistencia y la constante de tiempo deseada.

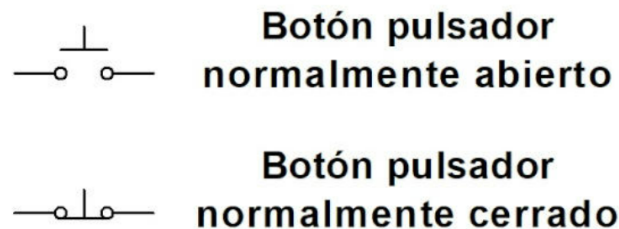


Figura 3: Símbolos esquemáticos de botones pulsadores: normalmente abierto y normalmente cerrado [1].

Como se muestra en la Figura 3, los símbolos permiten identificar fácilmente el tipo de pulsador según su configuración inicial.

2.5. Diseño

Para llevar a cabo la construcción de un dado electrónico, es necesario analizar el patrón que conforma cada una de sus caras. En este diseño se implementará una matriz de siete diodos LED distribuidos de tal forma que puedan representar cualquier número del 1 al 6 mediante una configuración específica de encendido.

Los patrones de iluminación serán los siguientes:

- **Número 1:** Se enciende únicamente el LED central.
- **Número 2:** Se activan dos LEDs opuestos en diagonal.
- **Número 3:** Se encienden el LED central y dos LEDs en diagonal.
- **Número 4:** Se activan los cuatro LEDs ubicados en las esquinas.
- **Número 5:** Se encienden los cuatro LEDs de las esquinas más el LED central.
- **Número 6:** Se activan todos los LEDs, excepto el central.

Para lograr estos patrones se utilizarán 7 LEDs distribuidos en grupos, los cuales se conectarán en serie de acuerdo con la lógica de encendido de cada número:

- Grupo 1: LED **a**.
- Grupo 2: LEDs **c** y **f**.
- Grupo 3: LEDs **b** y **g**.
- Grupo 4: LEDs **d** y **e**.

- Grupo 1: **GP1.**
- Grupo 2: **GP0.**
- Grupo 3: **GP4.**
- Grupo 4: **GP3.**
- Botón: **GP5.**

2.6. Cálculos de componentes

$$R = \frac{V_{OH} - n \cdot V_F}{I_F} \quad (1)$$

- Para el grupo 1 (con 2 LEDs): $R = 115 \, \Omega$
- Para los grupos 2, 3 y 4 (con 4 LEDs): $R = 15 \, \Omega$

En el caso del botón, se implementa un circuito de *pull-down* para asegurar que el pin del microcontrolador se mantenga en bajo cuando no está presionado. Usando una resistencia de 100 k Ω ,

la corriente es de solo 0,5 mA y la potencia disipada es de 0,0025 W, también dentro de los límites seguros.

Para evitar el efecto de rebote mecánico del botón, se añade un capacitor en paralelo con la resistencia de *pull-down*. Este forma un filtro RC cuya constante de tiempo se define como:

$$\tau = RC \quad (2)$$

Dado que no se especifica un valor exacto para τ , se elige un valor arbitrario de 1 ms. Con una resistencia de 10 k Ω , la capacitancia necesaria es:

$$C = \frac{1 \text{ ms}}{10 \text{ k}\Omega} = 100 \text{ nF} \quad (3)$$

2.7. Cotización

| Componente | Steren | Teltron | Notas |
|--------------------------|---------|------------------|-----------|
| PIC12f683 | | \$2.19 en Amazon | |
| Pulsador | ¢89 | | |
| LED | ¢59 | ¢79 | |
| Capacitor 0.1 μ F | ¢199 ud | ¢99 | |
| Resistencia 10 Ω | ¢25 ud | ¢249,00 | 1/2W; 5 % |
| Resistencia 15 Ω | ¢25 ud | ¢499,00 | 1/2W; 5 % |
| Resistencia 100 Ω | ¢25 ud | ¢499,00 | 1/2W; 5 % |

Tabla 1: Lista de componentes y precios (elaboración propia)

3. Análisis

3.1. Simulación

Con base en el desarrollo previo, se construyó el circuito final cuyo esquema se muestra en la figura 4. En dicho diagrama se observa que se han respetado las conexiones y los valores calculados para los componentes.

Es importante mencionar que, dado que una resistencia de 115 Ω no es un valor comercial común, se optó por dividirla en dos resistencias conectadas en serie: una de 100 Ω y otra de 15 Ω , ambas disponibles en el mercado y con tolerancia del 5 %. Esta configuración permite mantener el valor requerido con componentes fácilmente adquiribles.

3.2. Implementación

A continuación se describe el código desarrollado para controlar el dado electrónico. Se analizan sus funciones y estructura en secciones específicas para una mejor comprensión.

3.2.1. Configuración inicial y declaración de funciones

Listing 1: Configuración inicial y prototipos

```

1 #include <pic14/pic12f683.h>
2
3 // Configuraciones: Deshabilita MCLR y WDT
4 unsigned int __at 0x2007 __CONFIG = (_MCLRE_OFF & _WDTE_OFF);
5
6 // Prototipos de funciones
7 void generarAleatorio16(unsigned short *valorActual);
8 void retardoVisual(unsigned int duracion);

```

Esta sección incluye las instrucciones de configuración del microcontrolador. Se desactiva el pin MCLR (permitiendo que funcione como GPIO) y el Watchdog Timer (WDT). También se declaran los prototipos de funciones que se utilizarán más adelante en el programa.

3.2.2. Función main

Listing 2: Función principal

```

1 void main(void) {
2     TRISIO = 0b00100000; // GP5 como entrada; dem s como salida
3     GPIO = 0b00000000; // Inicializa pines en bajo
4
5     unsigned int retardo = 2000;
6     unsigned short semilla = 1;
7     unsigned short *ptrSemilla = &semilla;
8     unsigned short caraDado = 0;
9
10    while (1) {
11        if (GP5) {
12            switch (caraDado) {
13                case 1:
14                    GPIO = 0b000000010;
15                    retardoVisual(retardo);
16                    GPIO = 0b000000000;
17                    break;
18                case 2:
19                    GPIO = 0b000000001;
20                    retardoVisual(retardo);
21                    GPIO = 0b000000000;
22                    break;
23                case 3:
24                    GPIO = 0b000000011;
25                    retardoVisual(retardo);
26                    GPIO = 0b000000000;
27                    break;
28                case 4:
29                    GPIO = 0b00010001;
30                    retardoVisual(retardo);
31                    GPIO = 0b000000000;
32                    break;
33                case 5:
34                    GPIO = 0b00010110;
35                    retardoVisual(retardo);

```

```

36         GPIO = 0b00000000;
37         break;
38     case 6:
39         GPIO = 0b00011001;
40         retardoVisual(retardo);
41         GPIO = 0b00000000;
42         break;
43     default:
44         break;
45     }
46 } else {
47     GPIO = 0b00000000;
48     generarAleatorio16(ptrSemilla);
49     caraDado = 1 + (semilla % 6);
50 }
51 }
52 }

```

Esta es la función principal del programa. Inicializa los pines del microcontrolador y configura GP5 como entrada (botón). Luego, entra en un ciclo infinito donde se verifica si el botón fue presionado. Si es así, se encienden los LEDs correspondientes a la cara del dado almacenada en la variable 'caraDado'. En caso contrario, se apagan las salidas y se genera un nuevo número pseudoaleatorio para representar una nueva cara del dado.

3.2.3. Generador pseudoaleatorio generarAleatorio16

Listing 3: Función generadora pseudoaleatoria LFSR

```

1 void generarAleatorio16(unsigned short *valorActual) {
2     if ((*valorActual) & 1) {
3         (*valorActual) >>= 1;
4         (*valorActual) ^= (1 << 15) | (1 << 14) | (1 << 12) | (1 << 3);
5     } else {
6         (*valorActual) >>= 1;
7     }
8 }

```

Esta función implementa un generador de números pseudoaleatorios usando un registro de desplazamiento con retroalimentación lineal (LFSR) de 16 bits. Si el bit menos significativo es 1, se aplica una máscara XOR que define el polinomio generador; en caso contrario, solo se realiza el desplazamiento. Esto permite generar secuencias que simulan aleatoriedad con bajo consumo de recursos.

3.2.4. Función de retardo retardoVisual

Listing 4: Función de retardo basada en bucles

```

1 void retardoVisual(unsigned int duracion) {
2     unsigned int i, j;
3     for (i = 0; i < duracion; i++) {

```

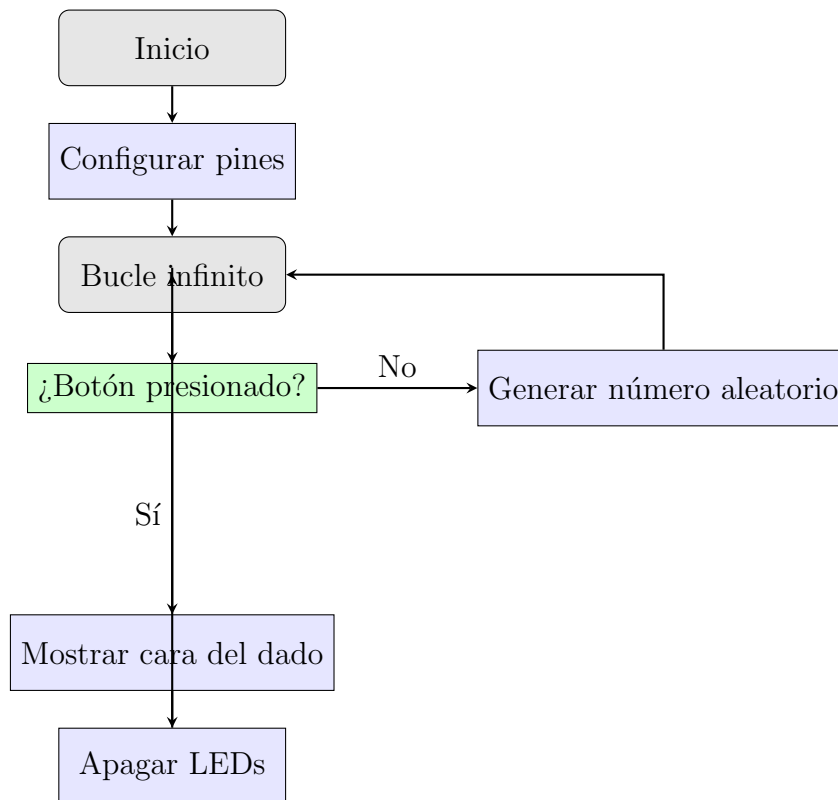


```
4     for (j = 0; j < 256; j++) {  
5         // Bucle vac o para generar retardo  
6     }  
7 }  
8 }
```

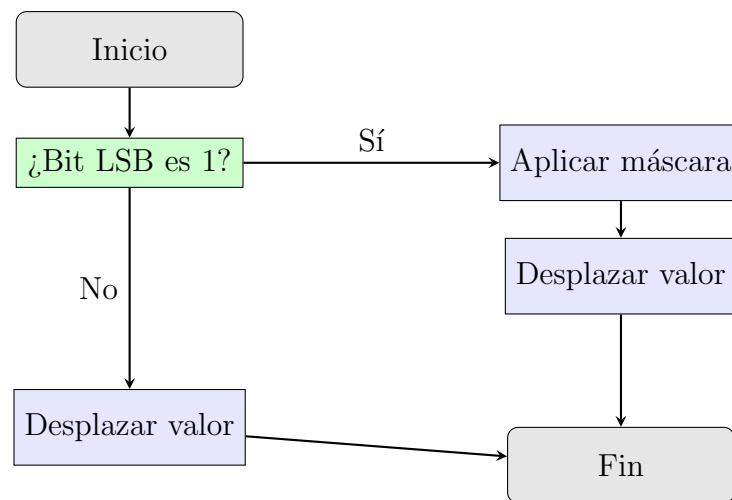
Esta función se utiliza para generar un retardo visual después de encender los LEDs, permitiendo que el número en el dado permanezca visible durante cierto tiempo. El retardo está basado en bucles vacíos y su duración depende del valor entregado en el parámetro 'duracion'.

3.3. diagrama de bloques

3.3.1. Funcion main



3.3.2. Función generarAleatorio16



4. Resultados

A continuación se muestran los resultados obtenidos tras la simulación del circuito del dado electrónico basado en el microcontrolador PIC12f683. Las imágenes corresponden a las salidas activadas de los LEDs para diferentes valores generados aleatoriamente por el programa. Cada figura representa una de las posibles caras del dado.

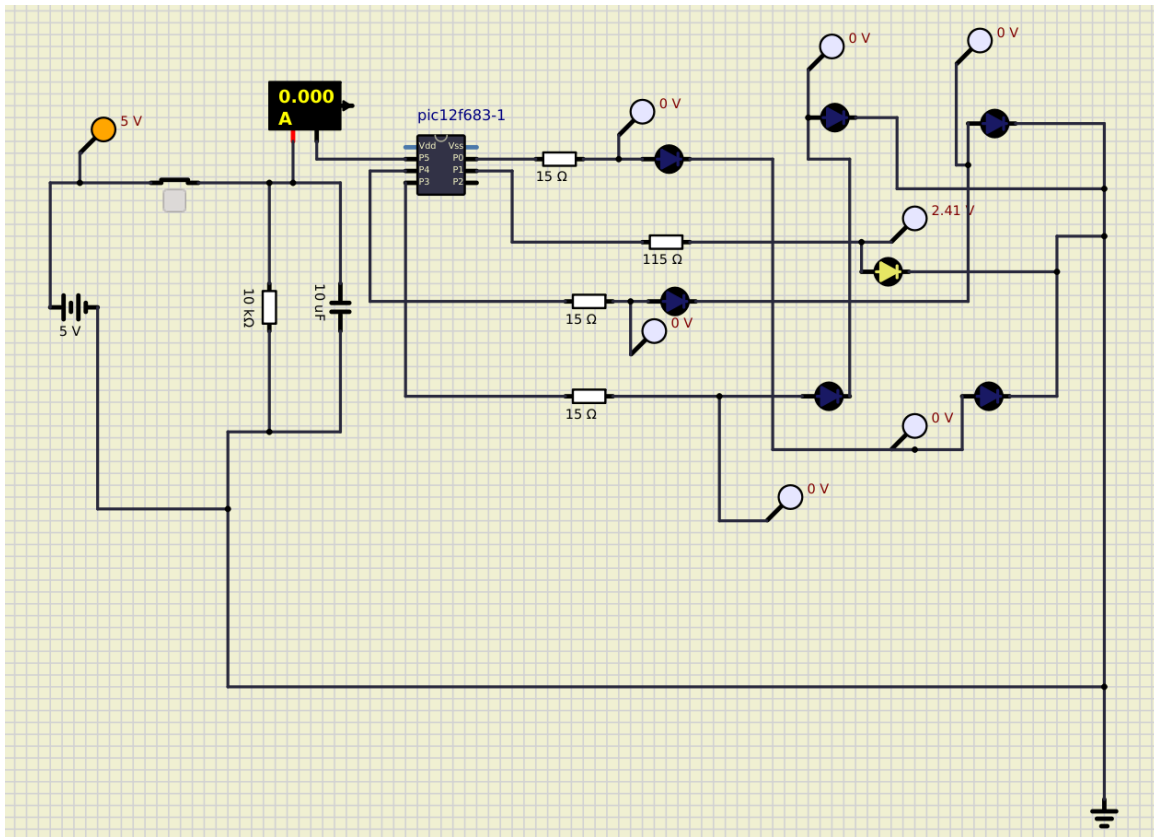


Figura 5: Resultado de la simulación para el número 1.

En la figura 5, se observa que solamente un LED está encendido en el centro del arreglo, indicando la cara número 1 del dado. El voltaje medido en este LED es de aproximadamente 2.4 V, lo que confirma que el diodo está polarizado directamente y operando en condiciones normales. El resto de los LEDs permanece en estado apagado (0 V), tal como se espera del diseño lógico codificado. Esto demuestra que el microcontrolador ha identificado correctamente la salida correspondiente a la cara número 1 del dado.

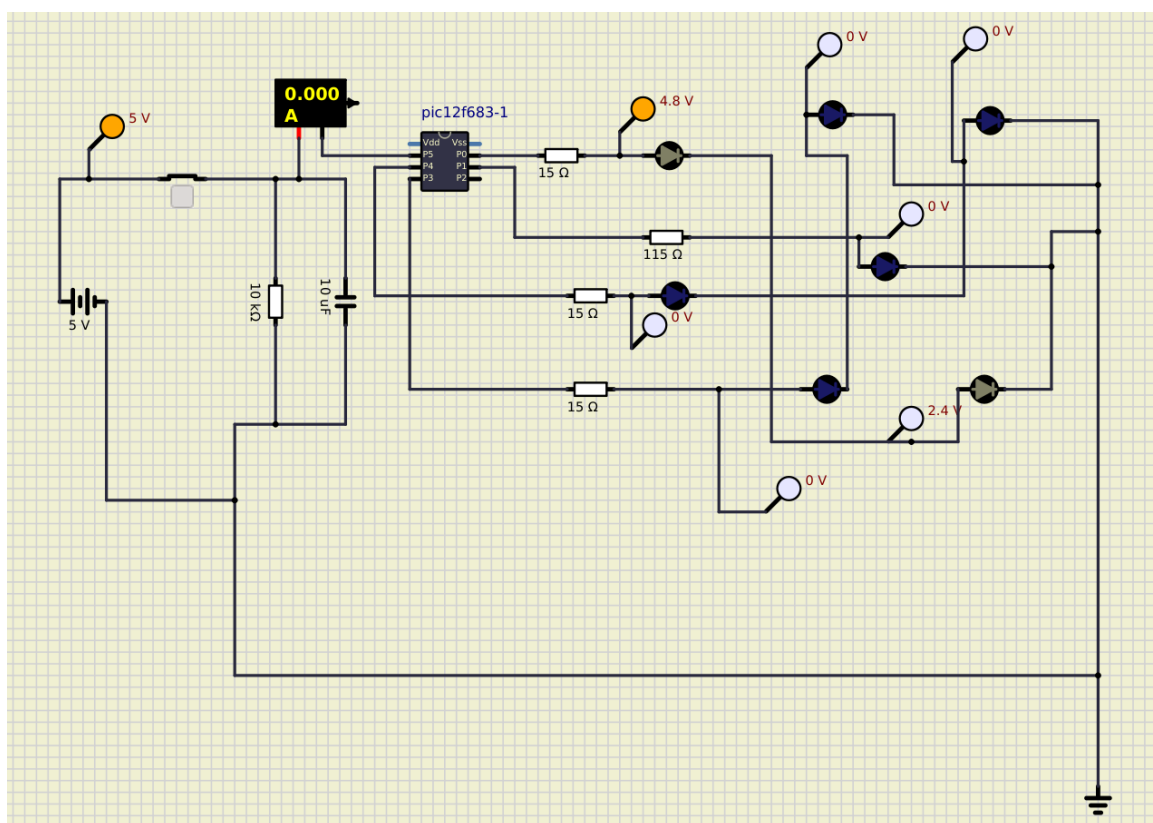


Figura 6: Resultado de la simulación para el número 2.

La figura 6 muestra dos LEDs activados ubicados en diagonal. Esta distribución corresponde a la cara número 2 del dado. El voltaje de activación se encuentra en torno a los 4.8 V en las salidas del microcontrolador, y se reduce a cerca de 2.4 V en los cátodos de los LEDs, lo que asegura un funcionamiento dentro del margen esperado. La coherencia entre la disposición espacial y la lógica implementada reafirma el correcto funcionamiento del sistema de control.

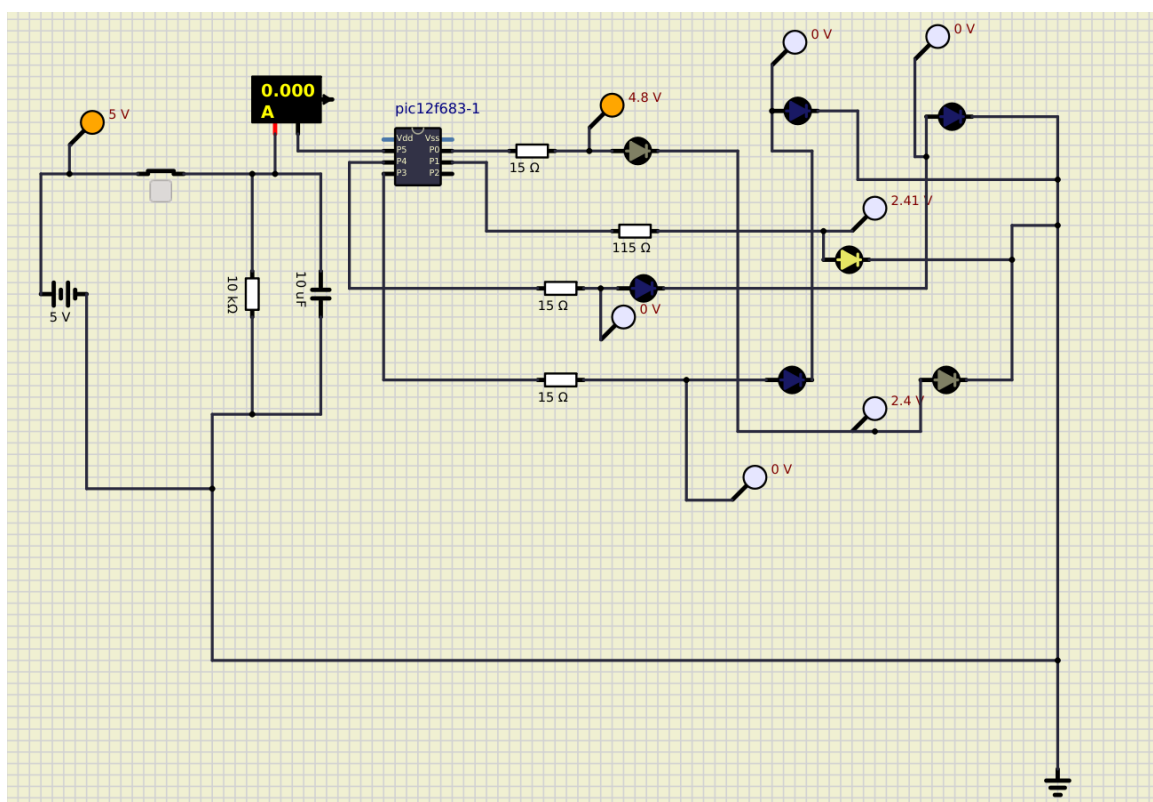


Figura 7: Resultado de la simulación para el número 3.

En la figura 7, se encienden tres LEDs: dos en diagonal y uno en el centro. Esta configuración representa la cara número 3 del dado. Es posible apreciar cómo los voltajes de activación siguen un patrón consistente, y los niveles de tensión en las resistencias de protección ($15\ \Omega$ y $115\ \Omega$) son adecuados para limitar la corriente sin afectar el brillo ni la integridad de los LEDs. El diseño cumple con el patrón esperado sin interferencia entre los canales.

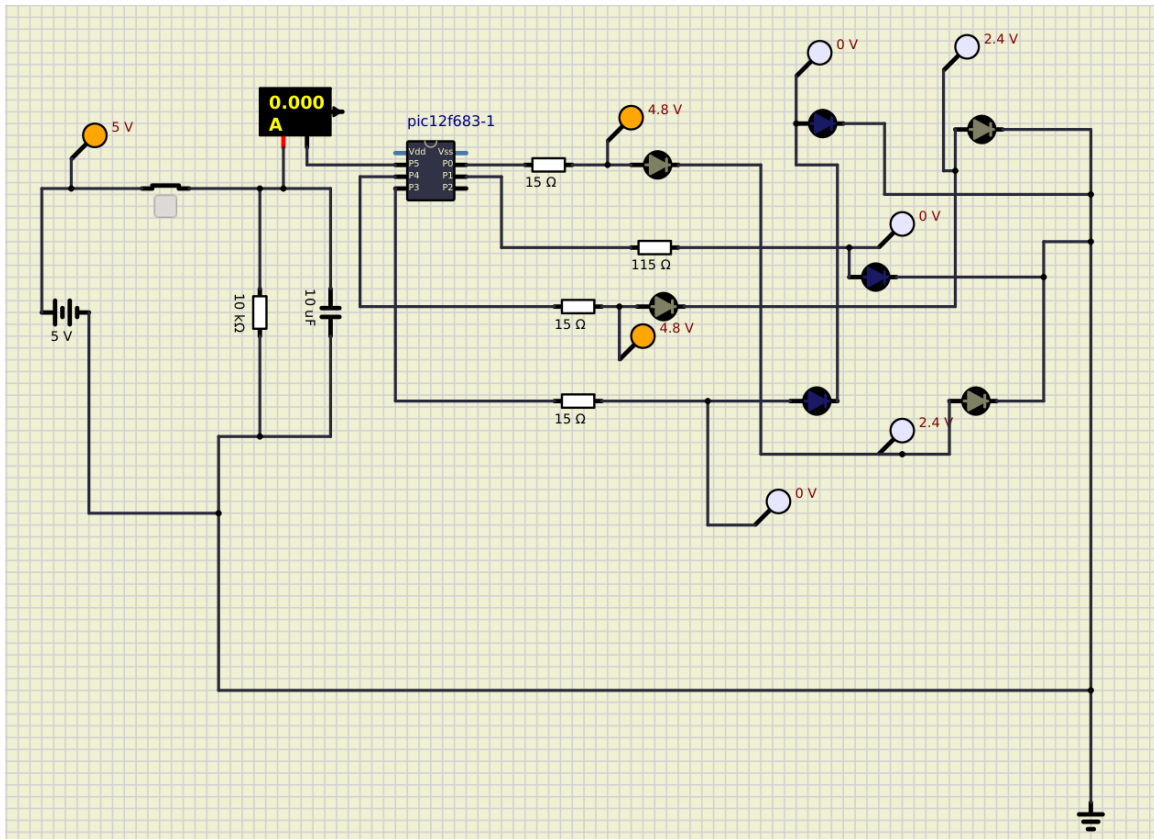


Figura 8: Resultado de la simulación para el número 4.

La figura 8 corresponde a la visualización de la cara número 4. Aquí se activan cuatro LEDs colocados en las esquinas del patrón. Este resultado valida tanto la lógica de agrupación de LEDs como la conexión en serie con resistencias que permiten limitar la corriente total en cada rama. Los voltajes observados indican que no hay caída excesiva ni sobrecorriente, lo cual es especialmente importante en configuraciones de múltiples salidas activas.

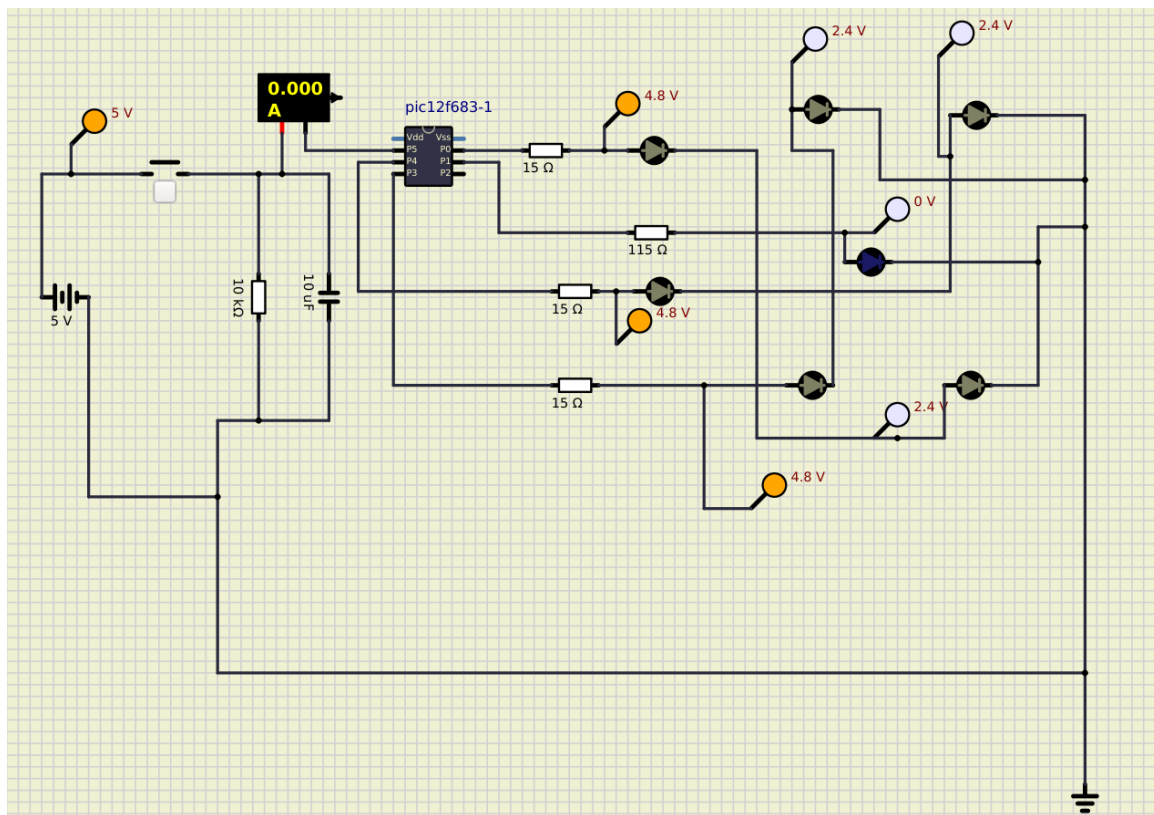


Figura 9: Resultado de la simulación para el número 6.

En la figura 9 se presenta el caso más complejo, correspondiente a la cara número 6 del dado. En esta simulación se activan seis LEDs: tres alineados verticalmente en cada lado del arreglo. Este patrón requiere la activación simultánea de todos los grupos de pines, excepto el LED central. Se puede verificar que los voltajes y caídas están dentro de lo previsto y que la alimentación general se mantiene estable, a pesar del número de elementos activos. Este resultado demuestra que el sistema está diseñado para manejar la carga combinada sin comprometer el rendimiento ni la integridad de los componentes.

5. Conclusiones

A lo largo del desarrollo de este proyecto se logró implementar con éxito un sistema electrónico capaz de simular el funcionamiento de un dado convencional, utilizando un microcontrolador PIC12f683 y una matriz de diodos LED. Desde el diseño del hardware hasta la programación del código embebido y la simulación final, se evidenció un dominio integral de conceptos de electrónica digital, programación en bajo nivel, y diseño lógico orientado a eventos físicos (como el uso de un botón).

Los resultados obtenidos a través de las simulaciones permiten concluir que la lógica de control implementada en el microcontrolador cumple correctamente su propósito. En cada una de las simulaciones, las salidas activas del sistema coincidieron con el patrón esperado de cada cara del dado (de 1 a 6), lo cual valida el funcionamiento del algoritmo pseudoaleatorio basado en LFSR, así como la precisión del control de salidas GPIO.

Se observó además que el sistema opera de forma eficiente en cuanto a voltajes y consumo de corriente. Los niveles de voltaje medidos en los LED encendidos se mantuvieron alrededor de 2.4V, lo que indica una polarización directa adecuada y sin sobrecargas. La inclusión de resistencias de protección calculadas específicamente para cada grupo de LEDs demostró ser efectiva, protegiendo los componentes sin comprometer la visibilidad del encendido.

El retardo visual implementado permite al usuario distinguir claramente qué número fue generado antes de apagar las salidas, lo que refuerza la percepción de funcionamiento similar al de un dado físico. Asimismo, la implementación del circuito de *pull-down* con una resistencia de 10 k Ω y un capacitor de 0,1 μ F resultó fundamental para estabilizar la entrada del botón, eliminando rebotes eléctricos y asegurando una respuesta confiable del sistema.

Desde el punto de vista de diseño práctico, el proyecto también destaca por su optimización de pines de salida. Al agrupar los LEDs estratégicamente y reducir la cantidad de GPIO necesarios, se demuestra una buena práctica de eficiencia en el uso de recursos limitados del microcontrolador.

En resumen, el dado electrónico diseñado y simulado se comporta de forma estable, confiable y predecible. Esto refleja no solo la validez del diseño electrónico, sino también la solidez de la lógica implementada y la correcta integración entre hardware y software.

6. Recomendaciones

Si bien el funcionamiento del sistema fue correcto en todos los escenarios simulados, existen varias oportunidades de mejora y expansión que podrían ser consideradas para futuras iteraciones del proyecto:

- **Incluir una fuente de entropía más fuerte para la generación aleatoria:** Aunque el generador LFSR ofrece buena dispersión para propósitos básicos, en aplicaciones reales donde se desee mayor aleatoriedad (como en juegos electrónicos), podría considerarse el uso de variaciones en el tiempo de pulsación del botón o ruido analógico como semilla adicional.
- **Optimizar el consumo energético:** Actualmente el circuito mantiene una corriente de consumo constante mientras los LEDs están encendidos. Implementar técnicas como apagado automático o reducción del tiempo de retardo podría hacer el sistema más eficiente en términos de energía, especialmente si se usara con baterías.
- **Agregar retroalimentación visual o sonora:** Para mejorar la experiencia del usuario, se puede integrar un pequeño buzzer o utilizar LEDs de diferentes colores o intensidades para simular mejor la tirada del dado. Incluso animaciones simples con secuencias de encendido podrían añadir dinamismo.
- **Utilizar encapsulado o interfaz física:** En una versión de uso real, se recomienda implementar el sistema en una carcasa o PCB impreso, con un botón externo accesible y una matriz de LEDs organizada de forma visualmente similar a un dado tradicional.
- **Compatibilidad con otros microcontroladores:** El diseño puede ser adaptado fácilmente a otras familias de microcontroladores como los ATtiny, PIC16 o incluso placas como Arduino, siempre que se respete la lógica de control. Esto puede facilitar su incorporación en plataformas educativas.

- **Documentación y modularidad del código:** Si se planea expandir el sistema (por ejemplo, para tirar múltiples dados o agregar más modos de juego), es recomendable organizar el código de manera más modular, separando la lógica de visualización, generación aleatoria y lectura del botón.
- **Proteger entradas sensibles:** Aunque el botón funciona correctamente, añadir una pequeña resistencia en serie con la entrada puede prevenir daños por descargas estáticas o cortocircuitos accidentales.
- **Agregar indicación de estado o inicio:** Un LED adicional o una animación de encendido podría ayudar a indicar al usuario que el sistema está listo para ser usado.

Con estas recomendaciones, el diseño podría volverse más robusto, escalable y adaptado a un entorno más amplio, manteniendo siempre la simplicidad que lo caracteriza como excelente proyecto de introducción a sistemas embebidos.

7. Apendice



PIC12F683

Data Sheet

8-Pin Flash-Based, 8-Bit
CMOS Microcontrollers with
nanoWatt 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.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELOQ, KEELOQ logo, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, PowerSmart, rfPIC, and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


AmpLab, FilterLab, Linear Active Thermistor, Migratable Memory, MXDEV, MXLAB, PS logo, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, PICkit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rfLAB, rfPICDEM, Select Mode, Smart Serial, SmartTel, Total Endurance, UNI/O, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2007, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona, Gresham, Oregon and Mountain View, California. 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 ==

8-Pin Flash-Based, 8-Bit CMOS Microcontrollers with nanoWatt Technology

High-Performance RISC CPU:

- Only 35 instructions to learn:
 - All single-cycle instructions except branches
- Operating speed:
 - DC – 20 MHz oscillator/clock input
 - DC – 200 ns instruction cycle
- Interrupt capability
- 8-level deep hardware stack
- Direct, Indirect and Relative Addressing modes

Special Microcontroller Features:

- Precision Internal Oscillator:
 - Factory calibrated to $\pm 1\%$, typical
 - Software selectable frequency range of 8 MHz to 125 kHz
 - Software tunable
 - Two-Speed Start-up mode
 - Crystal fail detect for critical applications
 - Clock mode switching during operation for power savings
- Power-Saving Sleep mode
- Wide operating voltage range (2.0V-5.5V)
- Industrial and Extended temperature range
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Brown-out Reset (BOR) with software control option
- Enhanced Low-Current Watchdog Timer (WDT) with on-chip oscillator (software selectable nominal 268 seconds with full prescaler) with software enable
- Multiplexed Master Clear with pull-up/input pin
- Programmable code protection
- High Endurance Flash/EEPROM cell:
 - 100,000 write Flash endurance
 - 1,000,000 write EEPROM endurance
 - Flash/Data EEPROM Retention: > 40 years

Low-Power Features:

- Standby Current:
 - 50 nA @ 2.0V, typical
- Operating Current:
 - 11 μ A @ 32 kHz, 2.0V, typical
 - 220 μ A @ 4 MHz, 2.0V, typical
- Watchdog Timer Current:
 - 1 μ A @ 2.0V, typical

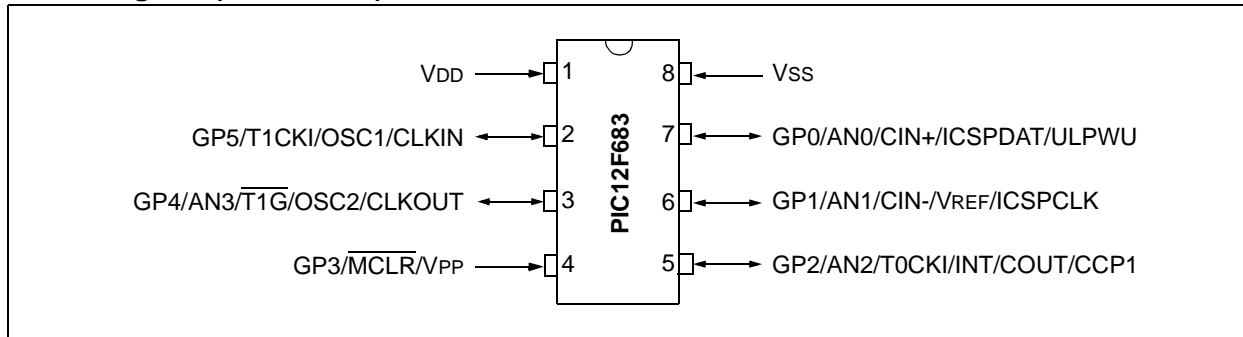
Peripheral Features:

- 6 I/O pins with individual direction control:
 - High current source/sink for direct LED drive
 - Interrupt-on-pin change
 - Individually programmable weak pull-ups
 - Ultra Low-Power Wake-up on GP0
- Analog Comparator module with:
 - One analog comparator
 - Programmable on-chip voltage reference (CVREF) module (% of VDD)
 - Comparator inputs and output externally accessible
- A/D Converter:
 - 10-bit resolution and 4 channels
- Timer0: 8-bit timer/counter with 8-bit programmable prescaler
- Enhanced Timer1:
 - 16-bit timer/counter with prescaler
 - External Timer1 Gate (count enable)
 - Option to use OSC1 and OSC2 in LP mode as Timer1 oscillator if INTOSC mode selected
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Capture, Compare, PWM module:
 - 16-bit Capture, max resolution 12.5 ns
 - Compare, max resolution 200 ns
 - 10-bit PWM, max frequency 20 kHz
- In-Circuit Serial Programming™ (ICSP™) via two pins

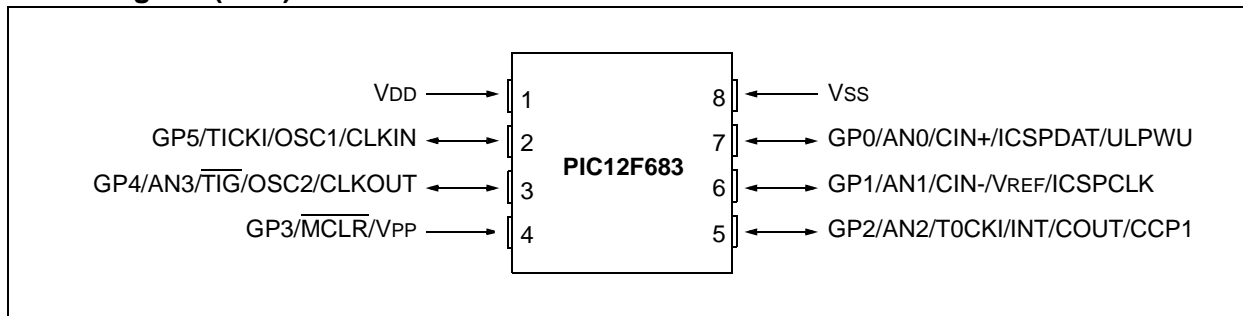
| Device | Program Memory | Data Memory | | I/O | 10-bit A/D (ch) | Comparators | Timers 8/16-bit |
|-----------|----------------|--------------|----------------|-----|-----------------|-------------|-----------------|
| | Flash (words) | SRAM (bytes) | EEPROM (bytes) | | | | |
| PIC12F683 | 2048 | 128 | 256 | 6 | 4 | 1 | 2/1 |

PIC12F683

8-Pin Diagram (PDIP, SOIC)



8-Pin Diagram (DFN)



8-Pin Diagram (DFN-S)

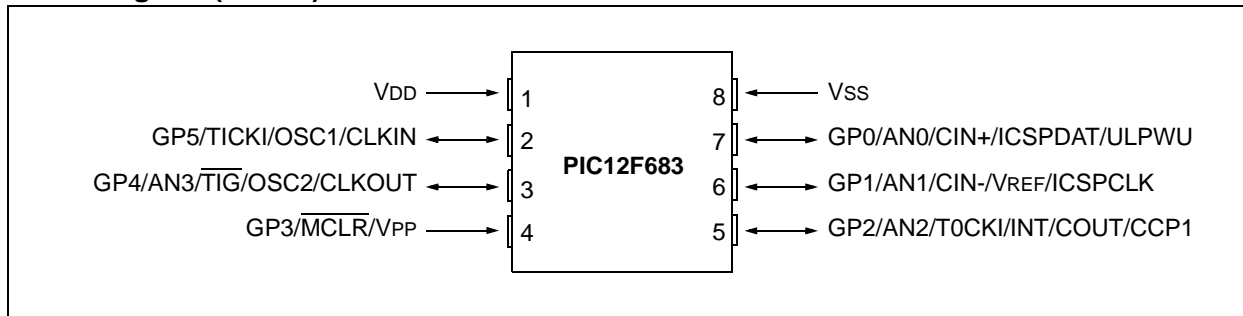


TABLE 1: 8-PIN SUMMARY

| I/O | Pin | Analog | Comparators | Timer | CCP | Interrupts | Pull-ups | Basic |
|--------------------|-----|----------|-------------|-------|------|------------|------------------|---------------|
| GP0 | 7 | AN0 | CIN+ | — | — | IOC | Y | ICSPDAT/ULPWU |
| GP1 | 6 | AN1/VREF | CIN- | — | — | IOC | Y | ICSPCLK |
| GP2 | 5 | AN2 | COU | T0CKI | CCP1 | INT/IOC | Y | — |
| GP3 ⁽¹⁾ | 4 | — | — | — | — | IOC | Y ⁽²⁾ | MCLR/VPP |
| GP4 | 3 | AN3 | — | T1G | — | IOC | Y | OSC2/CLKOUT |
| GP5 | 2 | — | — | T1CKI | — | IOC | Y | OSC1/CLKIN |
| — | 1 | — | — | — | — | — | — | VDD |
| — | 8 | — | — | — | — | — | — | VSS |

Note 1: Input only.

2: Only when pin is configured for external MCLR.

Table of Contents

| | | |
|------|---|-----|
| 1.0 | Device Overview | 5 |
| 2.0 | Memory Organization | 7 |
| 3.0 | Oscillator Module (With Fail-Safe Clock Monitor)..... | 19 |
| 4.0 | GPIO Port..... | 31 |
| 5.0 | Timer0 Module | 41 |
| 6.0 | Timer1 Module with Gate Control..... | 44 |
| 7.0 | Timer2 Module | 49 |
| 8.0 | Comparator Module..... | 51 |
| 9.0 | Analog-to-Digital Converter (ADC) Module | 61 |
| 10.0 | Data EEPROM Memory | 71 |
| 11.0 | Capture/Compare/PWM (CCP) Module | 75 |
| 12.0 | Special Features of the CPU | 83 |
| 13.0 | Instruction Set Summary | 101 |
| 14.0 | Development Support..... | 111 |
| 15.0 | Electrical Specifications..... | 115 |
| 16.0 | DC and AC Characteristics Graphs and Tables..... | 137 |
| 17.0 | Packaging Information..... | 159 |
| | Appendix A: Data Sheet Revision History..... | 165 |
| | Appendix B: Migrating From Other PIC® Devices | 165 |
| | The Microchip Web Site | 171 |
| | Customer Change Notification Service | 171 |
| | Customer Support | 171 |
| | Reader Response | 172 |
| | Product Identification System | 173 |

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@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)

When contacting a sales office, 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 to receive the most current information on all of our products.

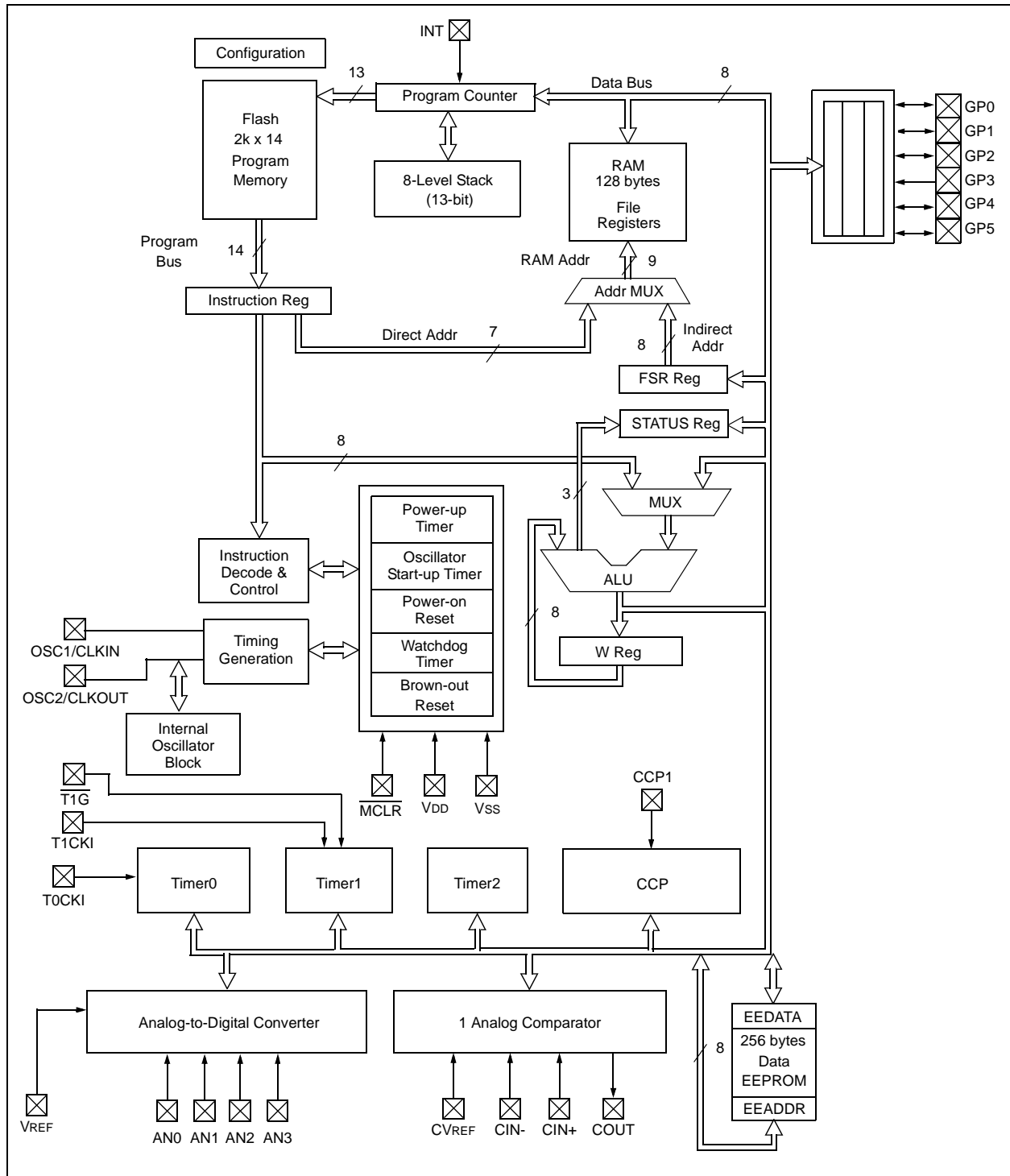
PIC12F683

NOTES:

1.0 DEVICE OVERVIEW

The PIC12F683 is covered by this data sheet. It is available in 8-pin PDIP, SOIC and DFN-S packages. Figure 1-1 shows a block diagram of the PIC12F683 device. Table 1-1 shows the pinout description.

FIGURE 1-1: PIC12F683 BLOCK DIAGRAM



PIC12F683

TABLE 1-1: PIC12F683 PINOUT DESCRIPTION

| Name | Function | Input Type | Output Type | Description |
|----------------------------|----------|------------|-------------|---|
| VDD | VDD | Power | — | Positive supply |
| GP5/T1CKI/OSC1/CLKIN | GP5 | TTL | CMOS | GPIO I/O with prog. pull-up and interrupt-on-change |
| | T1CKI | ST | — | Timer1 clock |
| | OSC1 | XTAL | — | Crystal/Resonator |
| | CLKIN | ST | — | External clock input/RC oscillator connection |
| GP4/AN3/T1G/OSC2/CLKOUT | GP4 | TTL | CMOS | GPIO I/O with prog. pull-up and interrupt-on-change |
| | AN3 | AN | — | A/D Channel 3 input |
| | T1G | ST | — | Timer1 gate |
| | OSC2 | — | XTAL | Crystal/Resonator |
| | CLKOUT | — | CMOS | Fosc/4 output |
| GP3/MCLR/VPP | GP3 | TTL | — | GPIO input with interrupt-on-change |
| | MCLR | ST | — | Master Clear with internal pull-up |
| | VPP | HV | — | Programming voltage |
| GP2/AN2/T0CKI/INT/COU/CCP1 | GP2 | ST | CMOS | GPIO I/O with prog. pull-up and interrupt-on-change |
| | AN2 | AN | — | A/D Channel 2 input |
| | T0CKI | ST | — | Timer0 clock input |
| | INT | ST | — | External Interrupt |
| | COU | — | CMOS | Comparator 1 output |
| | CCP1 | ST | CMOS | Capture input/Compare output/PWM output |
| GP1/AN1/CIN-/VREF/ICSPCLK | GP1 | TTL | CMOS | GPIO I/O with prog. pull-up and interrupt-on-change |
| | AN1 | AN | — | A/D Channel 1 input |
| | CIN- | AN | — | Comparator 1 input |
| | VREF | AN | — | External Voltage Reference for A/D |
| | ICSPCLK | ST | — | Serial Programming Clock |
| GP0/AN0/CIN+/ICSPDAT/ULPWU | GP0 | TTL | CMOS | GPIO I/O with prog. pull-up and interrupt-on-change |
| | AN0 | AN | — | A/D Channel 0 input |
| | CIN+ | AN | — | Comparator 1 input |
| | ICSPDAT | ST | CMOS | Serial Programming Data I/O |
| | ULPWU | AN | — | Ultra Low-Power Wake-up input |
| VSS | VSS | Power | — | Ground reference |

Legend:

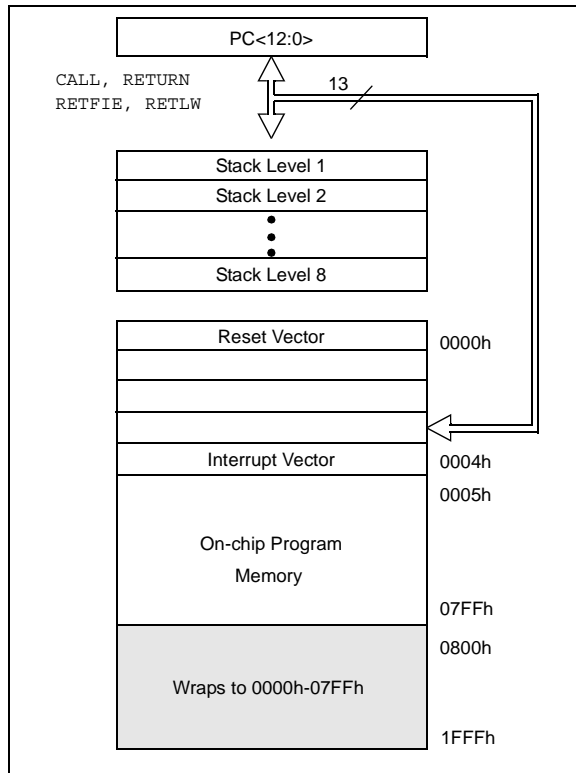
| | |
|-----------------------------|---|
| AN = Analog input or output | CMOS = CMOS compatible input or output |
| TTL = TTL compatible input | ST = Schmitt Trigger input with CMOS levels |
| HV = High Voltage | XTAL = Crystal |

2.0 MEMORY ORGANIZATION

2.1 Program Memory Organization

The PIC12F683 has a 13-bit program counter capable of addressing an 8k x 14 program memory space. Only the first 2k x 14 (0000h-07FFh) for the PIC12F683 is physically implemented. Accessing a location above these boundaries will cause a wraparound within the first 2K x 14 space. The Reset vector is at 0000h and the interrupt vector is at 0004h (see Figure 2-1).

FIGURE 2-1: PROGRAM MEMORY MAP AND STACK FOR THE PIC12F683



2.2 Data Memory Organization

The data memory (see Figure 2-2) is partitioned into two banks, which contain the General Purpose Registers (GPR) and the Special Function Registers (SFR). The Special Function Registers are located in the first 32 locations of each bank. Register locations 20h-7Fh in Bank 0 and A0h-BFh in Bank 1 are General Purpose Registers, implemented as static RAM. Register locations F0h-FFh in Bank 1 point to addresses 70h-7Fh in Bank 0. All other RAM is unimplemented and returns '0' when read. RP0 of the STATUS register is the bank select bit.


RP0

- 0 → Bank 0 is selected
- 1 → Bank 1 is selected

Note: The IRP and RP1 bits of the STATUS register are reserved and should always be maintained as '0's.

The special registers can be classified into two sets: core and peripheral. The Special Function Registers associated with the “core” are described in this section. Those related to the operation of the peripheral features are described in the section of that peripheral feature.

| File Address | | File Address | |
|---------------------------------------|-----|---------------------------------------|-----|
| Indirect addr. ⁽¹⁾ | 00h | Indirect addr. ⁽¹⁾ | 80h |
| TMR0 | 01h | OPTION_REG | 81h |
| PCL | 02h | PCL | 82h |
| STATUS | 03h | STATUS | 83h |
| FSR | 04h | FSR | 84h |
| GPIO | 05h | TRISIO | 85h |
| | 06h | | 86h |
| | 07h | | 87h |
| | 08h | | 88h |
| | 09h | | 89h |
| PCLATH | 0Ah | PCLATH | 8Ah |
| INTCON | 0Bh | INTCON | 8Bh |
| PIR1 | 0Ch | PIE1 | 8Ch |
| | 0Dh | | 8Dh |
| TMR1L | 0Eh | PCON | 8Eh |
| TMR1H | 0Fh | OSCCON | 8Fh |
| T1CON | 10h | OSCTUNE | 90h |
| TMR2 | 11h | | 91h |
| T2CON | 12h | PR2 | 92h |
| CCPR1L | 13h | | 93h |
| CCPR1H | 14h | | 94h |
| CCP1CON | 15h | WPU | 95h |
| | 16h | IOC | 96h |
| | 17h | | 97h |
| WDTCN | 18h | | 98h |
| CMCON0 | 19h | VRCON | 99h |
| CMCON1 | 1Ah | EEDAT | 9Ah |
| | 1Bh | EEADR | 9Bh |
| | 1Ch | EECON1 | 9Ch |
| | 1Dh | EECON2 ⁽¹⁾ | 9Dh |
| ADRESH | 1Eh | ADRESL | 9Eh |
| ADCON0 | 1Fh | ANSEL | 9Fh |
| General Purpose Registers 96 Bytes | 20h | General Purpose Registers 32 Bytes | A0h |
| | | | BFh |
| | | | C0h |
| | | | EFh |
| | | | F0h |
| | | Accesses 70h-7Fh | FFh |
| BANK 0 | | BANK 1 | |

 Unimplemented data memory locations, read as '0'.

Note 1: Not a physical register.

TABLE 2-1: PIC12F683 SPECIAL REGISTERS SUMMARY BANK 0

| Addr | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Page |
|--------|---------|--|--------------------|---------|--|-----------------|---------------------|---------|---------|-------------------|--------|
| Bank 0 | | | | | | | | | | | |
| 00h | INDF | Addressing this location uses contents of FSR to address data memory (not a physical register) | | | | | | | | xxxx xxxx | 17, 90 |
| 01h | TMR0 | Timer0 Module Register | | | | | | | | xxxx xxxx | 41, 90 |
| 02h | PCL | Program Counter's (PC) Least Significant Byte | | | | | | | | 0000 0000 | 17, 90 |
| 03h | STATUS | IRP ⁽¹⁾ | RP1 ⁽¹⁾ | RP0 | \overline{TO} | \overline{PD} | Z | DC | C | 0001 1xxx | 11, 90 |
| 04h | FSR | Indirect Data Memory Address Pointer | | | | | | | | xxxx xxxx | 17, 90 |
| 05h | GPIO | — | — | GP5 | GP4 | GP3 | GP2 | GP1 | GP0 | --xx xxxx | 31, 90 |
| 06h | — | Unimplemented | | | | | | | | — | — |
| 07h | — | Unimplemented | | | | | | | | — | — |
| 08h | — | Unimplemented | | | | | | | | — | — |
| 09h | — | Unimplemented | | | | | | | | — | — |
| 0Ah | PCLATH | — | — | — | Write Buffer for upper 5 bits of Program Counter | | | | --- | 0 0000 | 17, 90 |
| 0Bh | INTCON | GIE | PEIE | TOIE | INTE | GPIE | TOIF | INTF | GPIF | 0000 0000 | 13, 90 |
| 0Ch | PIR1 | EEIF | ADIF | CCP1IF | — | CMIF | OSFIF | TMR2IF | TMR1IF | 000- 0000 | 15, 90 |
| 0Dh | — | Unimplemented | | | | | | | | — | — |
| 0Eh | TMR1L | Holding Register for the Least Significant Byte of the 16-bit TMR1 | | | | | | | | xxxx xxxx | 44, 90 |
| 0Fh | TMR1H | Holding Register for the Most Significant Byte of the 16-bit TMR1 | | | | | | | | xxxx xxxx | 44, 90 |
| 10h | T1CON | T1GINV | TMR1GE | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{T1SYNC}$ | TMR1CS | TMR1ON | 0000 0000 | 47, 90 |
| 11h | TMR2 | Timer2 Module Register | | | | | | | | 0000 0000 | 49, 90 |
| 12h | T2CON | — | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | -000 0000 | 50, 90 |
| 13h | CCPR1L | Capture/Compare/PWM Register 1 Low Byte | | | | | | | | xxxx xxxx | 76, 90 |
| 14h | CCPR1H | Capture/Compare/PWM Register 1 High Byte | | | | | | | | xxxx xxxx | 76, 90 |
| 15h | CCP1CON | — | — | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 | --00 0000 | 75, 90 |
| 16h | — | Unimplemented | | | | | | | | — | — |
| 17h | — | Unimplemented | | | | | | | | — | — |
| 18h | WDTCON | — | — | — | WDTPS3 | WDTPS2 | WDTPS1 | WDTPS0 | SWDTEN | ---0 1000 | 97, 90 |
| 19h | CMCON0 | — | COUT | — | CINV | CIS | CM2 | CM1 | CM0 | -0-0 0000 | 56, 90 |
| 1Ah | CMCON1 | — | — | — | — | — | — | T1GSS | CMSYNC | ---- --10 | 57, 90 |
| 1Bh | — | Unimplemented | | | | | | | | — | — |
| 1Ch | — | Unimplemented | | | | | | | | — | — |
| 1Dh | — | Unimplemented | | | | | | | | — | — |
| 1Eh | ADRESH | Most Significant 8 bits of the left shifted A/D result or 2 bits of right shifted result | | | | | | | | xxxx xxxx | 61, 90 |
| 1Fh | ADCON0 | ADFM | VCFG | — | — | CHS1 | CHS0 | GO/DONE | ADON | 00-- 0000 | 65, 90 |

Legend: — = unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition,
shaded = unimplemented

Note 1: IRP and RP1 bits are reserved, always maintain these bits clear.

PIC12F683

TABLE 2-2: PIC12F683 SPECIAL FUNCTION REGISTERS SUMMARY BANK 1

| Addr | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Page |
|--------|--------------------|--|--------------------|---------|--|---------------------|---------|---------|---------|-------------------|--------|
| Bank 1 | | | | | | | | | | | |
| 80h | INDF | Addressing this location uses contents of FSR to address data memory (not a physical register) | | | | | | | | xxxx xxxx | 17, 90 |
| 81h | OPTION_REG | GPPU | INTEDG | T0CS | T0SE | PSA | PS2 | PS1 | PS0 | 1111 1111 | 12, 90 |
| 82h | PCL | Program Counter's (PC) Least Significant Byte | | | | | | | | 0000 0000 | 17, 90 |
| 83h | STATUS | IRP ⁽¹⁾ | RP1 ⁽¹⁾ | RP0 | T0 | PD | Z | DC | C | 0001 1xxx | 11, 90 |
| 84h | FSR | Indirect Data Memory Address Pointer | | | | | | | | xxxx xxxx | 17, 90 |
| 85h | TRISIO | — | — | TRISIO5 | TRISIO4 | TRISIO3 | TRISIO2 | TRISIO1 | TRISIO0 | --11 1111 | 32, 90 |
| 86h | — | Unimplemented | | | | | | | | — | — |
| 87h | — | Unimplemented | | | | | | | | — | — |
| 88h | — | Unimplemented | | | | | | | | — | — |
| 89h | — | Unimplemented | | | | | | | | — | — |
| 8Ah | PCLATH | — | — | — | Write Buffer for upper 5 bits of Program Counter | | | | --- | 0 0000 | 17, 90 |
| 8Bh | INTCON | GIE | PEIE | T0IE | INTE | GPIE | T0IF | INTF | GPIF | 0000 0000 | 13, 90 |
| 8Ch | PIE1 | EEIE | ADIE | CCP1IE | — | CMIE | OSFIE | TMR2IE | TMR1IE | 000- 0000 | 14, 90 |
| 8Dh | — | Unimplemented | | | | | | | | — | — |
| 8Eh | PCON | — | — | ULPWUE | SBOREN | — | — | POR | BOR | --01 --qq | 16, 90 |
| 8Fh | OSCCON | — | IRCF2 | IRCF1 | IRCF0 | OSTS ⁽²⁾ | HTS | LTS | SCS | -110 x000 | 20, 90 |
| 90h | OSCTUNE | — | — | — | TUN4 | TUN3 | TUN2 | TUN1 | TUN0 | ---0 0000 | 24, 90 |
| 91h | — | Unimplemented | | | | | | | | — | — |
| 92h | PR2 | Timer2 Module Period Register | | | | | | | | 1111 1111 | 49, 90 |
| 93h | — | Unimplemented | | | | | | | | — | — |
| 94h | — | Unimplemented | | | | | | | | — | — |
| 95h | WPU ⁽³⁾ | — | — | WPU5 | WPU4 | — | WPU2 | WPU1 | WPU0 | --11 -111 | 34, 90 |
| 96h | IOC | — | — | IOC5 | IOC4 | IOC3 | IOC2 | IOC1 | IOC0 | --00 0000 | 34, 90 |
| 97h | — | Unimplemented | | | | | | | | — | — |
| 98h | — | Unimplemented | | | | | | | | — | — |
| 99h | VRCON | VREN | — | VRR | — | VR3 | VR2 | VR1 | VR0 | 0-0- 0000 | 58, 90 |
| 9Ah | EEDAT | EEDAT7 | EEDAT6 | EEDAT5 | EEDAT4 | EEDAT3 | EEDAT2 | EEDAT1 | EEDAT0 | 0000 0000 | 71, 90 |
| 9Bh | EEADR | EEADR7 | EEADR6 | EEADR5 | EEADR4 | EEADR3 | EEADR2 | EEADR1 | EEADR0 | 0000 0000 | 71, 90 |
| 9Ch | EECON1 | — | — | — | — | WRERR | WREN | WR | RD | ---- x000 | 72, 91 |
| 9Dh | EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | | ---- ---- | 72, 91 |
| 9Eh | ADRESL | Least Significant 2 bits of the left shifted result or 8 bits of the right shifted result | | | | | | | | xxxx xxxx | 66, 91 |
| 9Fh | ANSEL | — | ADCS2 | ADCS1 | ADCS0 | ANS3 | ANS2 | ANS1 | ANS0 | -000 1111 | 33, 91 |

Legend: — = unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition, shaded = unimplemented

Note 1: IRP and RP1 bits are reserved, always maintain these bits clear.

2: OSTS bit of the OSCCON register reset to '0' with Dual Speed Start-up and LP, HS or XT selected as the oscillator.

3: GP3 pull-up is enabled when MCLRE is '1' in the Configuration Word register.

2.2.2.1 STATUS Register

The STATUS register, shown in Register 2-1, contains:

- Arithmetic status of the ALU
- Reset status
- Bank select bits for data memory (SRAM)

The STATUS register can be the destination for any instruction, like any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS`, will clear the upper three bits and set the Z bit. This leaves the STATUS register as `000u u1uu` (where u = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions are used to alter the STATUS register, because these instructions do not affect any Status bits. For other instructions not affecting any Status bits, see the "Instruction Set Summary".

Note 1: Bits IRP and RP1 of the STATUS register are not used by the PIC12F683 and should be maintained as clear. Use of these bits is not recommended, since this may affect upward compatibility with future products.

2: The C and DC bits operate as a Borrow and Digit Borrow out bit, respectively, in subtraction.

REGISTER 2-1: STATUS: STATUS REGISTER

| | | | | | | | |
|----------|----------|-------|------------------------|------------------------|-------|-------|-------|
| Reserved | Reserved | R/W-0 | R-1 | R-1 | R/W-x | R/W-x | R/W-x |
| IRP | RP1 | RP0 | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | Z | DC | C |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

| | |
|-------|--|
| bit 7 | IRP: This bit is reserved and should be maintained as '0' |
| bit 6 | RP1: This bit is reserved and should be maintained as '0' |
| bit 5 | RP0: Register Bank Select bit (used for direct addressing) 1 = Bank 1 (80h – FFh) 0 = Bank 0 (00h – 7Fh) |
| bit 4 | $\overline{\text{TO}}$: Time-out bit 1 = After power-up, <code>CLRWDT</code> instruction or <code>SLEEP</code> instruction 0 = A WDT time-out occurred |
| bit 3 | $\overline{\text{PD}}$: Power-down bit 1 = After power-up or by the <code>CLRWDT</code> instruction 0 = By execution of the <code>SLEEP</code> instruction |
| bit 2 | Z: Zero bit 1 = The result of an arithmetic or logic operation is zero 0 = The result of an arithmetic or logic operation is not zero |
| bit 1 | DC: Digit Carry/Borrow bit (<code>ADDWF</code> , <code>ADDLW</code> , <code>SUBLW</code> , <code>SUBWF</code> instructions), For Borrow, the polarity is reversed. 1 = A carry-out from the 4th low-order bit of the result occurred 0 = No carry-out from the 4th low-order bit of the result |
| bit 0 | C: Carry/Borrow bit ⁽¹⁾ (<code>ADDWF</code> , <code>ADDLW</code> , <code>SUBLW</code> , <code>SUBWF</code> instructions) 1 = A carry-out from the Most Significant bit of the result occurred 0 = No carry-out from the Most Significant bit of the result occurred |

Note 1: For Borrow, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high-order or low-order bit of the source register.

PIC12F683

2.2.2.2 OPTION Register

The OPTION register is a readable and writable register, which contains various control bits to configure:

- TMR0/WDT prescaler
- External GP2/INT interrupt
- TMR0
- Weak pull-ups on GPIO

Note: To achieve a 1:1 prescaler assignment for Timer0, assign the prescaler to the WDT by setting PSA bit of the OPTION register to '1'. See **Section 5.1.3 “Software Programmable Prescaler”**.

REGISTER 2-2: OPTION_REG: OPTION REGISTER

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|--------------------------|--------|-------|-------|-------|-------|-------|-------|
| $\overline{\text{GPPU}}$ | INTEDG | T0CS | T0SE | PSA | PS2 | PS1 | PS0 |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7 **$\overline{\text{GPPU}}$** : GPIO Pull-up Enable bit
1 = GPIO pull-ups are disabled
0 = GPIO pull-ups are enabled by individual PORT latch values in WPU register
- bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of INT pin
0 = Interrupt on falling edge of INT pin
- bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (Fosc/4)
- bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin
- bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module
- bit 2-0 **PS<2:0>**: Prescaler Rate Select bits

| BIT VALUE | TIMER0 RATE | WDT RATE |
|-----------|-------------|----------|
| 000 | 1 : 2 | 1 : 1 |
| 001 | 1 : 4 | 1 : 2 |
| 010 | 1 : 8 | 1 : 4 |
| 011 | 1 : 16 | 1 : 8 |
| 100 | 1 : 32 | 1 : 16 |
| 101 | 1 : 64 | 1 : 32 |
| 110 | 1 : 128 | 1 : 64 |
| 111 | 1 : 256 | 1 : 128 |

Note 1: A dedicated 16-bit WDT postscaler is available. See **Section 12.6 “Watchdog Timer (WDT)”** for more information.

2.2.2.3 INTCON Register

The INTCON register is a readable and writable register, which contains the various enable and flag bits for TMR0 register overflow, GPIO change and external GP2/INT pin interrupts.

Note: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE of the INTCON register. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 2-3: INTCON: INTERRUPT CONTROL REGISTER

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| GIE | PEIE | TOIE | INTE | GPIE | T0IF | INTF | GPIF |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|--|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared x = Bit is unknown |

- bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts
- bit 6 **PEIE:** Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
- bit 5 **TOIE:** Timer0 Overflow Interrupt Enable bit
1 = Enables the Timer0 interrupt
0 = Disables the Timer0 interrupt
- bit 4 **INTE:** GP2/INT External Interrupt Enable bit
1 = Enables the GP2/INT external interrupt
0 = Disables the GP2/INT external interrupt
- bit 3 **GPIE:** GPIO Change Interrupt Enable bit⁽¹⁾
1 = Enables the GPIO change interrupt
0 = Disables the GPIO change interrupt
- bit 2 **T0IF:** Timer0 Overflow Interrupt Flag bit⁽²⁾
1 = Timer0 register has overflowed (must be cleared in software)
0 = Timer0 register did not overflow
- bit 1 **INTF:** GP2/INT External Interrupt Flag bit
1 = The GP2/INT external interrupt occurred (must be cleared in software)
0 = The GP2/INT external interrupt did not occur
- bit 0 **GPIF:** GPIO Change Interrupt Flag bit
1 = When at least one of the GPIO <5:0> pins changed state (must be cleared in software)
0 = None of the GPIO <5:0> pins have changed state

Note 1: IOC register must also be enabled.

2: T0IF bit is set when TMR0 rolls over. TMR0 is unchanged on Reset and should be initialized before clearing T0IF bit.

PIC12F683

2.2.2.4 PIE1 Register

The PIE1 register contains the interrupt enable bits, as shown in Register 2-4.

Note: Bit PEIE of the INTCON register must be set to enable any peripheral interrupt.

REGISTER 2-4: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

| R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|-------|--------|-----|-------|-------|--------|--------|
| EEIE | ADIE | CCP1IE | — | CMIE | OSFIE | TMR2IE | TMR1IE |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7 **EEIE:** EE Write Complete Interrupt Enable bit
1 = Enables the EE write complete interrupt
0 = Disables the EE write complete interrupt
- bit 6 **ADIE:** A/D Converter (ADC) Interrupt Enable bit
1 = Enables the ADC interrupt
0 = Disables the ADC interrupt
- bit 5 **CCP1IE:** CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt
- bit 4 **Unimplemented:** Read as '0'
- bit 3 **CMIE:** Comparator Interrupt Enable bit
1 = Enables the Comparator 1 interrupt
0 = Disables the Comparator 1 interrupt
- bit 2 **OSFIE:** Oscillator Fail Interrupt Enable bit
1 = Enables the oscillator fail interrupt
0 = Disables the oscillator fail interrupt
- bit 1 **TMR2IE:** Timer2 to PR2 Match Interrupt Enable bit
1 = Enables the Timer2 to PR2 match interrupt
0 = Disables the Timer2 to PR2 match interrupt
- bit 0 **TMR1IE:** Timer1 Overflow Interrupt Enable bit
1 = Enables the Timer1 overflow interrupt
0 = Disables the Timer1 overflow interrupt

2.2.2.5 PIR1 Register

The PIR1 register contains the interrupt flag bits, as shown in Register 2-5.

Note: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE of the INTCON register. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 2-5: PIR1: PERIPHERAL INTERRUPT REQUEST REGISTER 1

| | | | | | | | |
|-------|-------|--------|-----|-------|-------|--------|--------|
| R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| EEIF | ADIF | CCP1IF | — | CMIF | OSFIF | TMR2IF | TMR1IF |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit
-n = Value at POR

W = Writable bit
'1' = Bit is set

U = Unimplemented bit, read as '0'
'0' = Bit is cleared
x = Bit is unknown

- bit 7 **EEIF:** EEPROM Write Operation Interrupt Flag bit
1 = The write operation completed (must be cleared in software)
0 = The write operation has not completed or has not been started
- bit 6 **ADIF:** A/D Interrupt Flag bit
1 = A/D conversion complete
0 = A/D conversion has not completed or has not been started
- bit 5 **CCP1IF:** CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode
- bit 4 **Unimplemented:** Read as '0'
- bit 3 **CMIF:** Comparator Interrupt Flag bit
1 = Comparator 1 output has changed (must be cleared in software)
0 = Comparator 1 output has not changed
- bit 2 **OSFIF:** Oscillator Fail Interrupt Flag bit
1 = System oscillator failed, clock input has changed to INTOSC (must be cleared in software)
0 = System clock operating
- bit 1 **TMR2IF:** Timer2 to PR2 Match Interrupt Flag bit
1 = Timer2 to PR2 match occurred (must be cleared in software)
0 = Timer2 to PR2 match has not occurred
- bit 0 **TMR1IF:** Timer1 Overflow Interrupt Flag bit
1 = Timer1 register overflowed (must be cleared in software)
0 = Timer1 has not overflowed

PIC12F683

2.2.2.6 PCON Register

The Power Control (PCON) register contains flag bits (see Table 12-2) to differentiate between a:

- Power-on Reset ($\overline{\text{POR}}$)
- Brown-out Reset ($\overline{\text{BOR}}$)
- Watchdog Timer Reset (WDT)
- External MCLR Reset

The PCON register also controls the Ultra Low-Power Wake-up and software enable of the $\overline{\text{BOR}}$.

The PCON register bits are shown in Register 2-6.

REGISTER 2-6: PCON: POWER CONTROL REGISTER

| U-0 | U-0 | R/W-0 | R/W-1 | U-0 | U-0 | R/W-0 | R/W-x |
|-------|-----|--------|--------|-----|-----|-------------------------|-------------------------|
| — | — | ULPWUE | SBOREN | — | — | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **ULPWUE:** Ultra Low-Power Wake-Up Enable bit

1 = Ultra Low-Power Wake-up enabled

0 = Ultra Low-Power Wake-up disabled

bit 4 **SBOREN:** Software BOR Enable bit⁽¹⁾

1 = BOR enabled

0 = BOR disabled

bit 3-2 **Unimplemented:** Read as '0'

bit 1 **POR:** Power-on Reset Status bit

1 = No Power-on Reset occurred

0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0 **BOR:** Brown-out Reset Status bit

1 = No Brown-out Reset occurred

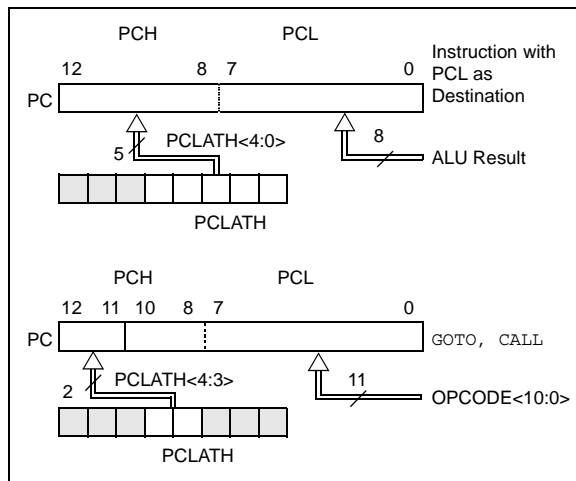
0 = A Brown-out Reset occurred (must be set in software after a Power-on Reset or Brown-out Reset occurs)

Note 1: Set BOREN<1:0> = 01 in the Configuration Word register for this bit to control the $\overline{\text{BOR}}$.

2.3 PCL and PCLATH

The Program Counter (PC) is 13 bits wide. The low byte comes from the PCL register, which is a readable and writable register. The high byte (PC<12:8>) is not directly readable or writable and comes from PCLATH. On any Reset, the PC is cleared. Figure 2-3 shows the two situations for the loading of the PC. The upper example in Figure 2-3 shows how the PC is loaded on a write to PCL (PCLATH<4:0> → PCH). The lower example in Figure 2-3 shows how the PC is loaded during a CALL or GOTO instruction (PCLATH<4:3> → PCH).

FIGURE 2-3: LOADING OF PC IN DIFFERENT SITUATIONS



2.3.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL). When performing a table read using a computed GOTO method, care should be exercised if the table location crosses a PCL memory boundary (each 256-byte block). Refer to the Application Note AN556, "Implementing a Table Read" (DS00556).

2.3.2 STACK

The PIC12F683 family has an 8-level x 13-bit wide hardware stack (see Figure 2-1). The stack space is not part of either program or data space and the Stack Pointer is not readable or writable. The PC is PUSHed onto the stack when a CALL instruction is executed or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not affected by a PUSH or POP operation.

The stack operates as a circular buffer. This means that after the stack has been PUSHed eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on).

Note 1: There are no Status bits to indicate stack overflow or stack underflow conditions.

2: There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, RETURN, RETLW and RETFIE instructions or the vectoring to an interrupt address.

2.4 Indirect Addressing, INDF and FSR Registers

The INDF register is not a physical register. Addressing the INDF register will cause indirect addressing.

Indirect addressing is possible by using the INDF register. Any instruction using the INDF register actually accesses data pointed to by the File Select Register (FSR). Reading INDF itself indirectly will produce 00h. Writing to the INDF register indirectly results in a no operation (although Status bits may be affected). An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit of the STATUS register, as shown in Figure 2-4.

A simple program to clear RAM location 20h-2Fh using indirect addressing is shown in Example 2-1.

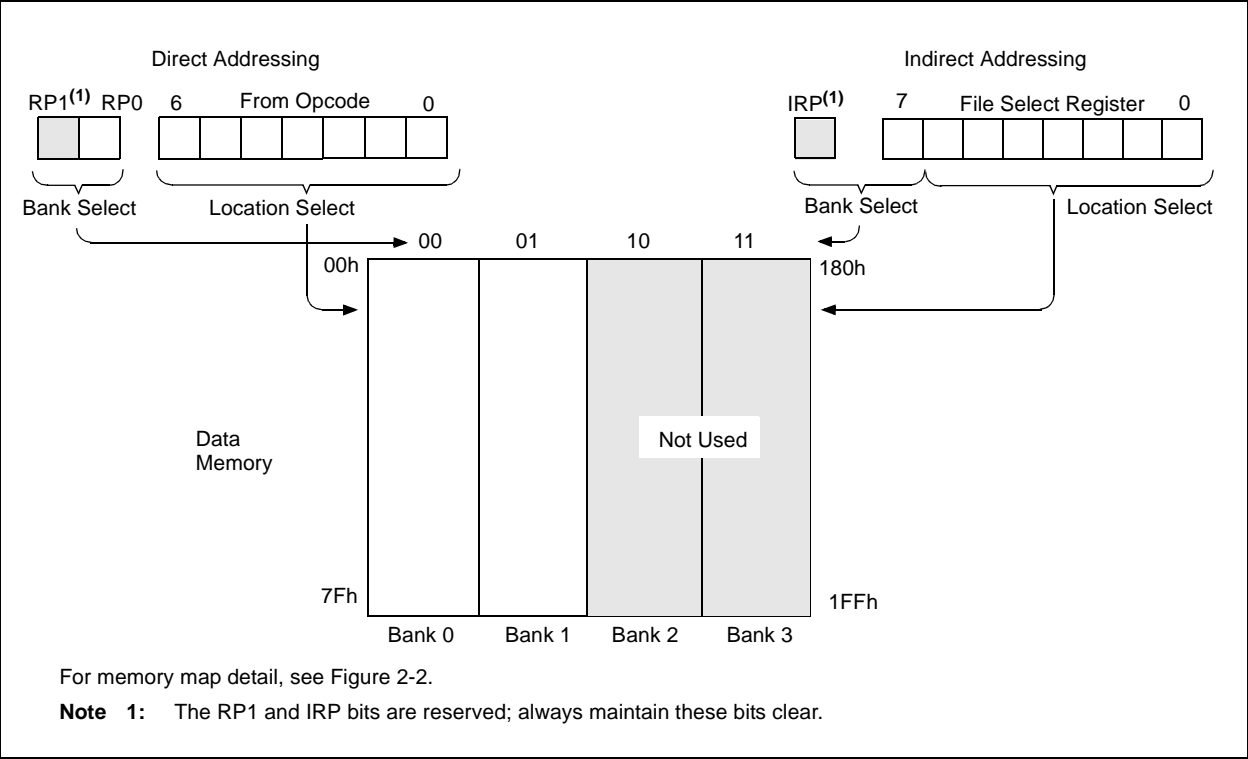
EXAMPLE 2-1: INDIRECT ADDRESSING

```

MOV LW    0x20    ;initialize pointer
MOV WF    FSR     ;to RAM
NEXT      CLRF    INDF ;clear INDF register
          INCF    FSR ;inc pointer
          BTFSS   FSR,4 ;all done?
          GOTO    NEXT ;no clear next
CONTINUE  ;yes continue
    
```

PIC12F683

FIGURE 2-4: DIRECT/INDIRECT ADDRESSING PIC12F683



Para el resto del datasheet visite [4]

Referencias

- [1] Electricidad Básica. *Botones pulsadores: Qué son, tipos y cómo funcionan*. Accedido: 25-mar-2025. 2024. URL: <https://electricidad-basica.com/dispositivos-electricos/botones-pulsadores/>.
- [2] IBM. *¿Qué es un microcontrolador?* Accedido: 25-mar-2025. 2024. URL: <https://www.ibm.com/mx-es/think/topics/microcontroller>.
- [3] Ielectel. *¿Qué es un resistor y cómo funciona?* Accedido: 25-mar-2025. 2024. URL: <https://ielectel.com/que-es-un-resistor-y-como-funciona/>.
- [4] Microchip Technology Inc. *PIC12F683 - 8-bit PIC Microcontroller with 8-bit ADC, Comparator, EUSART, Capture Compare PWM*. Accedido: 25-mar-2025. 2024. URL: <https://www.microchip.com/en-us/product/pic12f683>.
- [5] Raymundo Pizano. *Arduino 5: Mi primer código. Encender un LED, resistencias y la Ley de Ohm*. Accedido: 25-mar-2025. 2024. URL: <https://raymundopizano.com/blog/arduino/arduino-5-mi-primer-codigo-encender-un-led-resistencias-y-la-ley-de-ohm/>.