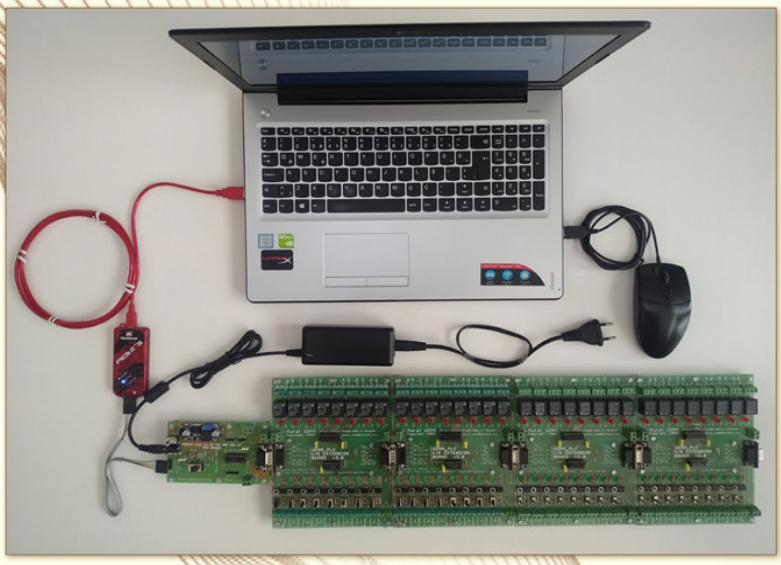
PIC16F1847 MICROCONTROLLER-BASED PROGRAMMABLE LOGIC CONTROLLER

HARDWARE AND BASIC CONCEPTS



MURAT UZAM

CRC Press
Taylor & Francis Group

PIC16F1847 Microcontroller-Based Programmable Logic Controller

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

PIC16F1847 Microcontroller-Based Programmable Logic Controller

Hardware and Basic Concepts

Murat Uzam

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

First edition published 2021

by CRC Press
6000 Broken Sound Parkway NW, Suite 300, Boca Raton, FL 33487-2742
and by CRC Press
2 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN

First edition published by CRC Press 2021

CRC Press is an imprint of Taylor & Francis Group, LLC

ISBN: 9780367506391 (hbk)
ISBN: 9781003050605 (ebk)

Typeset in Times
by Deanta Global Publishing Services, Chennai, India

Visit the Routledge website: https://www.routledge.com/9780367506391

To the memory of my beloved father, Mehmet Uzam (1937–2017)
to my mother Zeynep Uzam
to my family
who love and support me
and
to my teachers and students
who enriched my knowledge

Contents

Prologue

Think globally, act locally.

Never give up.

No pain, no gain.

Practice makes perfect.

If we hear, we forget
If we see, we remember
If we do, we understand.

Success is not an accident, excellence is not a coincidence.

Think out of the box.

Preface

Programmable Logic Controllers (PLC) have been extensively used in industry for the past five decades. PLC manufacturers offer different PLCs in terms of functions, program memories, and the number of inputs/outputs (I/O), ranging from a few to thousands of I/Os. The design and implementation of PLCs have long been a secret of the PLC manufacturers. A serious project was reported by the author of this book in his previous book, entitled *Building a Programmable Logic Controller with a PIC16F648A Microcontroller*, published by CRC Press in 2014, to describe a microcontroller-based implementation of a PLC. The current project, called *PIC16F1847 Microcontroller-Based Programmable Logic Controller*, is based on the improved version of the project reported in the above-mentioned book. The improvements include both hardware and software elements. The current project is reported in three books and a downloadable document explaining application examples:

1. *PIC16F1847 Microcontroller-Based Programmable Logic Controller: Hardware and Basic Concepts* (this book)
2. *PIC16F1847 Microcontroller-Based Programmable Logic Controller: Intermediate Concepts*
3. *PIC16F1847 Microcontroller-Based Programmable Logic Controller: Advanced Concepts*

The current project is presented for students attending the related departments of engineering or technology faculties, for practicing engineers, and for hobbyists who want to learn how to design and use a microcontroller-based PLC. The book assumes the reader has taken courses on digital logic design, microcontrollers, and PLCs. In addition, the reader is expected to be familiar with the PIC16F series of microcontrollers and to have been exposed to writing programs using PIC assembly language within the MPLAB integrated development environment.

The contents of this book may be used to construct two different courses. The first one may involve teaching the use of the PLC technology as described in this book. This course may well fit in the related departments of both engineering and technology faculties. The second one may involve teaching how to design the PLC technology. This second course may be taught in electrical and electronics engineering and computer engineering departments.

Source and example files defined for the basic concepts of the PIC16F1847-Based PLC project are downloadable from this book's webpage under the downloads section.

In addition, PCB files of the CPU and I/O extension boards of the PIC16F1847-Based PLC can also be downloaded from the same link.

Prof. Dr. Murat UZAM
Yozgat Bozok Üniversitesi
Mühendislik-Mimarlık Fakültesi
Elektrik-Elektronik Mühendisliği Bölümü
Yozgat
Turkey

About the Author

Murat Uzam was born in Söke, Turkey, in 1968. He received his B.Sc. and M.Sc. degrees from the Electrical Engineering Department, Yıldız Technical University, İstanbul, Turkey, in 1989 and 1991, respectively, and his Ph.D. degree from the University of Salford, Salford, U.K., in 1998. He was with Niğde University, Turkey, from 1993 to 2010 in the Department of Electrical and Electronics Engineering as a Research Assistant, Assistant Professor, Associate Professor, and Professor. He was a Professor in the Department of Electrical and Electronics Engineering at Melikşah University in Kayseri, Turkey, from 2011 to 2016. Since 15 April 2020, he has been serving as a Professor in the Department of Electrical and Electronics Engineering at Yozgat Bozok University in Yozgat, Turkey.

He was a Visiting Researcher with INRIA, University of Metz and University of Rennes, France, in 1999, with the University of Toronto, Toronto, ON, Canada, in 2003, and with Xidian University, Xi'an, China, in 2013, 2015, and 2019.

He has published 47 conference papers and 106 journal and magazine papers, 70 of which are indexed by Science Citation Index Expanded (SCIE). He has published two books in Turkish and five books in English by CRC Press (Taylor & Francis Group). According to Publons, his H-Index is 16 and his papers have been cited more than 1370 times by the papers indexed in the SCIE. Dr. Uzam has been serving as a reviewer for prestigious journals and conferences. According to Publons, the number of his verified reviews is 70. His current research interests include design and implementation of discrete event control systems modelled by Petri nets and, in particular, deadlock prevention/liveness enforcing in flexible manufacturing systems, programmable logic controllers (PLCs), microcontrollers (especially PIC microcontrollers), and the design of microcontroller-based PLCs. The details of his studies are accessible from his web page: https://pbs.bozok.edu.tr/goster.php?lookup=1074

Background and Use of the Book

This project has been completed during the search for an answer to the following question: "How could one design and implement a programmable logic controller (PLC)?". An answer to this question was provided by the author in his previous book entitled *Building a Programmable Logic Controller with a PIC16F648A Microcontroller*, published by CRC Press in 2014. This project is based on the improved version of the PLC project reported in the above-mentioned book. So many new features have been included within the PIC16F1847-Based PLC project to make it an almost perfect PLC. The reader should be aware of the fact that this project does not include a graphical interface PC software as in commercial PLCs for developing PLC programs. Rather, PLC programs are developed by using macros as done in the Instruction List (IL) PLC programming language. An interested and skilled reader could well (and is encouraged to) develop a graphical interface PC software for easy use of the PIC16F1847-Based PLC.

The improvements of the PLC project reported in this book (*Hardware and Basic Concepts*) compared with the previous version are summarized as follows.

1. The current version of the PLC explained in this book is based on the PIC16F1847 microcontroller with: 8,192 words of flash program memory, 1,024 bytes of SRAM data memory, 256 bytes of EEPROM data memory, the maximum operating speed of 32 MHz, a 16-level-deep hardware stack, and an enhanced instruction set consisting of 49 single-word instructions, while the previous one was based on the PIC16F648A microcontroller with: 4,096 words of flash program memory, 256 bytes of SRAM data memory, 256 bytes of EEPROM data memory, the maximum operating speed of 20 MHz, an 8-level-deep hardware stack, and an instruction set consisting of 35 single-word instructions.
2. The hardware explained in this book consists of 1 CPU board and 4 digital I/O extension boards, while the previous one consisted of 1 CPU board and 2 digital I/O extension boards.
3. The clock frequency is 32 MHz in the current version of PLC, while it was 20 MHz in the previous version.
4. The current version of the PLC supports up to 32 digital inputs and 32 digital outputs, while the previous one supported 16 digital inputs and 16 digital outputs.
5. The current version of the PLC supports up to 4 analog inputs and 1 analog output, while the previous one did not support analog inputs/outputs.
6. The current version of the PLC supports 1,024 internal relays (memory bits), while the previous one supported only 32 internal relays.

7. The current version of the PLC provides 30 contact and relay-based instructions (macros), while the previous version provided 18 contact and relay-based instructions.
8. The current version of the PLC provides 14 flip-flop instructions (macros), while the previous version provided 8 flip-flop instructions.
9. The current version of the PLC provides 80 timers in total. These timers can be chosen from on-delay timers (TON_8 or TON_16), retentive on-delay timers (RTO_8 or RTO_16), off-delay timers (TOF_8 or TOF_16), pulse timers (TP_8 or TP_16), extended pulse timers (TEP_8 or TEP_16), and oscillator timers (TOS_8 or TOS_16). The timers with the suffix "_8" have 8-bit resolution, i.e., they are based on 8-bit registers, while the timers with the suffix "_16" have 16-bit resolution, i.e., they are based on 16-bit registers. On the other hand, the previous version of the PLC provided 8 on-delay timers (TON_8), 8 off-delay timers (TOF_8), 8 pulse timers (TP_8), and 8 oscillator timers (TOS_8). All these timers had 8-bit resolution, i.e., they were based on 8-bit registers.
10. The current version of the PLC provides 80 counters in total. These counters can be chosen from up counters (CTU_8 or CTU_16), down counters (CTD_8 or CTD_16), up/down counters (CTUD_8 or CTUD_16), and generalized up/down counters (GCTUD_8 or GCTUD_16). The counters with the suffix "_8" have 8-bit resolution, i.e., they are based on 8-bit registers, while the counters with the suffix "_16" have 16-bit resolution, i.e., they are based on 16-bit registers. On the other hand, the previous version of the PLC provided in total only 8 counters (CTU8 or CTD8 or CTUD8). They had 8-bit resolution, i.e., they were based on 8-bit registers.
11. The current version of the PLC provides 30 comparison instructions (macros), while the previous version provided 12 comparison instructions.
12. Almost all macros are improved compared with the previous versions, in terms of flexibility. For example, there is no restriction on the SRAM Banks, i.e., Boolean variables, 8-bit variables, and 16-bit variables used as a parameter in an instruction can be in any Bank. This was not the case in the previous version.
13. Flowcharts are provided to help the understanding of macros (instructions).

In order to follow the topics explained in this book properly, it is expected that the reader will construct his/her own PIC16F1847-Based PLC consisting of the CPU board and 4 I/O extension boards using the PCB files provided on the book's webpage under the downloads section. In addition, the reader should also download and make use of the PLC project files from the book's webpage. In this project, as the PIC Assembly is used as the programming language within the MPLAB integrated development environment (IDE), the reader is referred to the homepage of Microchip (http://www.microchip.com/) to obtain the latest version of MPLAB IDE. References [R1 and R2] may be useful to understand some aspects of the PIC16F1847 microcontroller and MPASM™ Assembler, respectively.

The contents of this book's 7 chapters are explained briefly, as follows.

1. **Hardware of the PIC16F1847-Based PLC:** In this chapter, the hardware structure of the PIC16F1847-Based PLC, consisting of 32 discrete inputs,

32 discrete outputs, 4 analog inputs, 1 analog output, and 2 PWM outputs is explained in detail.

2. **Basic Software:** This chapter explains the basic software structure of the PIC16F1847-Based PLC. A PLC scan cycle includes the following: (1) obtain the inputs, (2) run the user program, (3) update the outputs. In addition, it is also necessary to define and initialize all variables used within a PLC. Necessary functions are all described as PIC Assembly macros to be used in the PIC16F1847-Based PLC. The source files of the PIC16F1847-Based PLC are as follows: "PICPLC_PIC16F1847_memory.inc" (the individual bits of 8-bit SRAM registers M0, M1, …, M127 are defined in this file), "PICPLC_PIC16F1847_main.asm" (processor-specific variable definitions, PICPLC definitions, the user program, and subroutines are included in the project by using this file), "PICPLC_PIC16F1847_user.inc" (this file contains two macros, namely "user_program_1" and "user_program_2", in order to accommodate user programs), "PICPLC_PIC16F1847_subr.inc" (this file contains the "subroutines" macro and it is defined to obtain time delays at the expense of CPU clocks; the "subroutines" macro contains two time delay–related subroutines: "pause_1ms" and "pause_10us"), and "PICPLC_PIC16F1847_macros.inc". The file "PICPLC_PIC16F1847_macros.inc" contains the following macros: "initialize" (for PLC initialization), "ISR" (interrupt service routines), "get_inputs" (for handling the inputs), "lpf_progs" (low-pass digital filter macros for analog inputs), and "send_outputs" (for sending the outputs).
3. **Contact and Relay-Based Macros:** The following contact and relay-based macros are described in this chapter: "ld" (load), "ld_not" (load_not), "not", "or", "or_not", "nor", "and", "and_not", "nand", "xor", "xor_not", "xnor", "out", "out_not", "mid_out" (midline output), "mid_out_not" (inverted midline output), "in_out", "inv_out", "_set", "_reset", "SR" (set–reset), "RS" (reset–set), "r_edge" (rising edge detector), "f_edge" (falling edge detector), "r_toggle" (output toggle with rising edge detector), "f_toggle" (output toggle with falling edge detector), "adrs_re" (Address rising edge detector), "adrs_fe" (Address falling edge detector), "setBF" (set bit field), and "resetBF" (reset bit field). These macros are defined to operate on 1-bit (Boolean) variables.
4. **Flip-Flop Macros:** The following flip-flop macros are described in this chapter: "latch1" (D latch with active high enable), "latch0" (D latch with active low enable), "dff_r" (rising edge–triggered D flip-flop), "dff_r_SR" (rising edge–triggered D flip-flop with active high preset [S] and clear [R] inputs), "dff_f" (falling edge–triggered D flip-flop), "dff_f_SR" (falling edge–triggered D flip-flop with active high preset [S] and clear [R] inputs), "tff_r" (rising edge–triggered T flip-flop), "tff_r_SR" (rising edge–triggered T flip-flop with active high preset [S] and clear [R] inputs), "tff_f" (falling edge–triggered T flip-flop), "tff_f_SR" (falling edge–triggered T flip-flop with active high preset [S] and clear [R] inputs), "jkff_r" (rising edge–triggered JK flip-flop), "jkff_r_SR" (rising edge–triggered JK flip-flop with active high preset [S] and clear [R] inputs), "jkff_f" (falling

edge–triggered JK flip-flop), and "jkff_f_SR" (falling edge–triggered JK flip-flop with active high preset [S] and clear [R] inputs). Flip-flop macros are defined to operate on Boolean (1-bit) variables. 21 examples are provided to show the applications of these flip-flop macros, including the implementation of asynchronous and synchronous counters, and shift registers constructed by using the flip-flop macros.

5. **Timer Macros:** The following timer macros are described in this chapter: "TON_8" (8-bit on-delay timer), "TON_16" (16-bit on-delay timer), "RTO_8" (8-bit retentive on-delay timer), "RTO_16" (16-bit retentive on-delay timer), "TOF_8" (8-bit off-delay timer), "TOF_16" (16-bit off-delay timer), "TP_8" (8-bit pulse timer), "TP_16" (16-bit pulse timer), "TEP_8" (8-bit extended pulse timer), "TEP_16" (16-bit extended pulse timer), "TOS_8" (8-bit oscillator timer), and "TOS_16" (16-bit oscillator timer).
6. **Counter Macros:** The following counter macros are described in this chapter: "CTU_8" (8-bit up counter), "CTU_16" (16-bit up counter), "CTD_8" (8-bit down counter), "CTD_16" (16-bit down counter), "CTUD_8" (8-bit up/down counter), "CTUD_16" (8-bit up/down counter), "GCTUD_8" (8-bit generalized up/down counter), and "GCTUD_16" (16-bit generalized up/down counter).
7. **Comparison Macros:** In this chapter, the majority of the comparison macros are described according to the following notation: **GT** (**G**reater **T**han—"$>$"), **GE** (**G**reater than or **E**qual to—"$\geq$"), **EQ** (**EQ**ual to—"$=$"), **LT** (**L**ess **T**han—"$<$"), **LE** (**L**ess than or **E**qual to—"$\leq$"), or **NE** (**N**ot **E**qual to—"$\neq$"). The contents of two 8-bit registers (R1 and R2) are compared with the following comparison macros: "R1_GT_R2" (Is R1 greater than R2?), "R1_GE_R2" (Is R1 greater than or equal to R2?), "R1_EQ_R2" (Is R1 equal to R2?), "R1_LT_R2" (Is R1 less than R2?), "R1_LE_R2" (Is R1 less than or equal to R2?), and "R1_NE_R2" (Is R1 not equal to R2?). Similar comparison macros are also described for comparing the contents of an 8-bit register (R) with an 8-bit constant (K): "R_GT_K" (Is R greater than K?), "R_GE_K" (Is R greater than or equal to K?), "R_EQ_K" (Is R equal to K?), "R_LT_K" (Is R less than K?), "R_LE_K" (Is R less than or equal to K?), and "R_NE_K" (Is R not equal to K?).

   The contents of two 16-bit registers (R1 and R2) are compared with the following comparison macros: "R1_GT_R2_16" (Is R1 greater than R2?), "R1_GE_R2_16" (Is R1 greater than or equal to R2?), "R1_EQ_R2_16" (Is R1 equal to R2?), "R1_LT_R2_16" (Is R1 less than R2?), "R1_LE_R2_16" (Is R1 less than or equal to R2?), and "R1_NE_R2_16" (Is R1 not equal to R2?). Similar comparison macros are also described for comparing the contents of a 16-bit register (R) with a 16-bit constant (K): "R_GT_K_16" (Is R greater than K?), "R_GE_K_16" (Is R greater than or equal to K?), "R_EQ_K_16" (Is R equal to K?), "R_LT_K_16" (Is R less than K?), "R_LE_K_16" (Is R less than or equal to K?), and "R_NE_K_16" (Is R not equal to K?). In addition, the following comparison macros are also provided: "in_RANGE" (Is the value within the given range?), "in_RANGE_16" (Is the value within the given range?), "out_RANGE" (Is the value out of the given range?),

TABLE 1
General Characteristics of the PIC16F1847-Based PLC

Inputs/Outputs/Functions	Byte addresses/Related bytes	Bit addresses or function names/numbers
32 discrete inputs	I0,	I0.0, I0.1, …, I0.7
(external inputs: 5 or 24V DC)	I1,	I1.0, I1.1, …, I1.7
	I2,	I2.0, I2.1, …, I2.7
	I3	I3.0, I3.1, …, I3.7
32 discrete outputs	Q0,	Q0.0, Q0.1, …, Q0.7
(relay-type outputs)	Q1,	Q1.0, Q1.1, …, Q1.7
	Q2,	Q2.0, Q2.1, …, Q2.7
	Q3	Q3.0, Q3.1, …, Q3.7
4 analog inputs	AI0H:AI0L,	AI0H,1, AI0H,0, …, AI0L,0
	AI1H:AI1L,	AI1H,1, AI1H,0, …, AI1L,0
	AI2H:AI2L,	AI2H,1, AI2H,0, …, AI2L,0
	AI3H:AI3L	AI3H,1, AI3H,0, …, AI3L,0
1 analog output	-	RA2
1 high speed counter input	-	RB6
2 PWM outputs	-	RA4 & RA7
Drum sequencer instruction with up to 16 steps and 16 outputs on each step	Details are available in Chapter 4 of Volume III - *Advanced Concepts*	Details are available in Chapter 4 of Volume III - *Advanced Concepts*
1,024 internal relays	M0,	M0.0, M0.1, …, M0.7
(memory bits)	M1,	M1.0, M1.1, …, M1.7
	.	.
	.	.
	M127	M127.0, M127.1, …, M127.7
80 8-bit on-delay timers	TV_L, TV_L+1, …, TV_L+79	TQ0, TQ1, …, TQ79
(TON_8)	T_Q0, T_Q1, …, T_Q9	
80 8-bit retentive on-delay timers	TV_L, TV_L+1, …, TV_L+79	TQ0, TQ1, …, TQ79
(RTO_8)	T_Q0, T_Q1, …, T_Q9	
80 8-bit off-delay timers	TV_L, TV_L+1, …, TV_L+79	TQ0, TQ1, …, TQ79
(TOF_8)	T_Q0, T_Q1, …, T_Q9	
80 8-bit pulse timers	TV_L, TV_L+1, …, TV_L+79	TQ0, TQ1, …, TQ79
(TP_8)	T_Q0, T_Q1, …, T_Q9	
80 8-bit extended pulse timers	TV_L, TV_L+1, …, TV_L+79	TQ0, TQ1, …, TQ79
(TEP_8)	T_Q0, T_Q1, …, T_Q9	
80 8-bit oscillator timers	TV_L, TV_L+1, …, TV_L+79	TQ0, TQ1, …, TQ79
(TOS_8)	T_Q0, T_Q1, …, T_Q9	
80 16-bit on-delay timers	TV_L, TV_L+1, …, TV_L+79	TQ0, TQ1, …, TQ79
(TON_16)	TV_H, TV_H+1, …, TV_H+79	
	T_Q0, T_Q1, …, T_Q9	

(Continued)

TABLE 1 (CONTINUED)
General Characteristics of the PIC16F1847-Based PLC

Inputs/Outputs/Functions	Byte addresses/Related bytes	Bit addresses or function names/numbers
80 16-bit retentive on-delay timers (RTO_16)	TV_L, TV_L+1, …, TV_L+79 TV_H, TV_H+1, …, TV_H+79 T_Q0, T_Q1, …, T_Q9	TQ0, TQ1, …, TQ79
80 16-bit off-delay timers (TOF_16)	TV_L, TV_L+1, …, TV_L+79 TV_H, TV_H+1, …, TV_H+79 T_Q0, T_Q1, …, T_Q9	TQ0, TQ1, …, TQ79
80 16-bit pulse timers (TP_16)	TV_L, TV_L+1, …, TV_L+79 TV_H, TV_H+1, …, TV_H+79 T_Q0, T_Q1, …, T_Q9	TQ0, TQ1, …, TQ79
80 16-bit extended pulse timers (TEP_16)	TV_L, TV_L+1, …, TV_L+79 TV_H, TV_H+1, …, TV_H+79 T_Q0, T_Q1, …, T_Q9	TQ0, TQ1, …, TQ79
80 16-bit oscillator timers (TOS_16)	TV_L, TV_L+1, …, TV_L+79 TV_H, TV_H+1, …, TV_H+79 T_Q0, T_Q1, …, T_Q9	TQ0, TQ1, …, TQ79
80 8-bit up counters (CTU_8)	CV_L, CV_L+1, …, CV_L+79 C_Q0, C_Q1, …, C_Q9	CQ0, CQ1, …, CQ79
80 8-bit down counters (CTD_8)	CV_L, CV_L+1, …, CV_L+79 C_Q0, C_Q1, …, C_Q9	CQ0, CQ1, …, CQ79
80 8-bit up/down counters (CTUD_8)	CV_L, CV_L+1, …, CV_L+79 C_Q0, C_Q1, …, C_Q9	CQ0, CQ1, …, CQ79
80 8-bit generalized up/down counters (GCTUD_8)	CV_L, CV_L+1, …, CV_L+79 C_Q0, C_Q1, …, C_Q9	CQ0, CQ1, …, CQ79
80 16-bit up counters (CTU_16)	CV_L, CV_L+1, …, CV_L+79 CV_H, CV_H+1, …, CV_H+79 C_Q0, C_Q1, …, C_Q9	CQ0, CQ1, …, CQ79
80 16-bit down counters (CTD_16)	CV_L, CV_L+1, …, CV_L+79 CV_H, CV_H+1, …, CV_H+79 C_Q0, C_Q1, …, C_Q9	CQ0, CQ1, …, CQ79
80 16-bit up/down counters (CTUD_16)	CV_L, CV_L+1, …, CV_L+79 CV_H, CV_H+1, …, CV_H+79 C_Q0, C_Q1, …, C_Q9	CQ0, CQ1, …, CQ79
80 16-bit generalized up/down counters (GCTUD_16)	CV_L, CV_L+1, …, CV_L+79 CV_H, CV_H+1, …, CV_H+79 C_Q0, C_Q1, …, C_Q9	CQ0, CQ1, …, CQ79

"out_RANGE_16" (Is the value out of the given range?), "Hbit_CaC" (high bit count and compare), and "Lbit_CaC" (low bit count and compare). Note that this chapter is provided as downloadable ancillary material.

Application Examples: In total there are 20 application examples considered. For some application examples, more than one solution is provided in order to point out

how different methods can be used for controlling the same problem. When the three books are purchased separately, application examples 1–9 (or 10–11 and 13–18; 7–12 and 20, respectively) are provided as downloadable ancillary material for the book *PIC16F1847 Microcontroller-Based Programmable Logic Controller: Hardware and Basic Concepts* (*Intermediate Concepts*; *Advanced Concepts*, respectively). On the other hand, when the three books are purchased as a set, all application examples are provided as a single ancillary material.

Appendix A: The list of components for all boards and modules developed in this project as reported in this book, together with the photographs of all components, are provided in Appendix A.

Table 1 shows the general characteristics of the PIC16F1847-Based PLC.

IMPORTANT NOTES

1. At any time, at most 80 different timers can be used. A unique timer number from 0 to 79 can be assigned to only one of the macros "TP_8", "TEP_8", "TOS_8", "TON_16", "RTO_16", "TOF_16", "TP_16", "TEP_16", and "TOS_16".
2. At any time, at most 80 different counters can be used. A unique counter number from 0 to 79 can be assigned to only one of the macros "CTU_8", "CTD_8", "CTUD_8", "GCTUD_8", "CTU_16", "CTD_16", "CTUD_16", and "GCTUD_16".

REFERENCES

R1. PIC16(L)F1847 Data Sheet, DS40001453F, 2011–2017, Microchip Technology Inc. http://ww1.microchip.com/downloads/en/DeviceDoc/40001453F.pdf

R2. MPASM™ Assembler, MPLINK™ Object Linker, MPLIB™ Object Librarian User's Guide DS33014J, 2005, Microchip Technology Inc. http://ww1.microchip.com/downloads/en/devicedoc/33014j.pdf

1 Hardware of the PIC16F1847-Based PLC

The hardware of the PIC16F1847-Based PLC consists of mainly two parts: the *CPU board* and the *I/O extension board*. The schematic diagram and the photograph of the PIC16F1847-Based PLC CPU board are shown in Figures 1.1 and 1.2, respectively. The CPU board contains mainly *three sections*: power, programming, and CPU (central processing unit).

The **power section** accepts 12V DC input used as the operating voltage of relays. 5V DC is also used for ICs, inputs, etc. An adjustable LM2596 step-down voltage regulator module is used to obtain 5V DC voltage from the 12V DC input voltage. It has the following specifications—conversion efficiency: up to 92%; switching frequency: 150 KHz; rectifier: nonsynchronous rectification; module properties: non-isolated step-down module (buck); operating temperature: industrial grade (–40 to +85); load regulation: ± 0.5%; voltage regulation: ± 2.5%; dynamic response speed: 5% 200 μs; input voltage: 3–40V; output voltage: 1.5–35V (adjustable); output current: maximum 3A; size: 43mm*21mm*14mm (length*width*height).

It is important to note that the output voltage (OUT+) of the adjustable LM2596 step-down voltage regulator module must be set to 5.00V by adjusting the potentiometer on the module before inserting the CPU. 12V DC input voltage can be subjected to electric surge or electrostatic discharge on the external terminal connections. The TVS (transient voltage suppressor) 1.5KE13A shown in the circuit provides highly effective protection against such discharges. It is also used to protect the circuit from accidental reverse polarity of the DC input voltage. For a proper operation of the PIC16F1847-Based PLC make sure that the DC input voltage < 13V DC.

The **programming section** deals with the programming of the PIC16F1847 microcontroller. For programming the PIC16F1847 in circuit, it is necessary to use a PIC programmer hardware and a software with ICSP (in-circuit serial programming) capability. In this project, Microchip's PICkit 3 In-Circuit Debugger/Programmer (www.microchip.com/PICkit3) is used as the PIC programmer hardware. MPLAB X IDE software (www.microchip.com/mplab/mplab-x-ide), freely available by Microchip (www.microchip.com), is used for the program development and for programming the PIC16F1847 microcontroller. The ICSP connector takes the lines VPP (MCLR), VDD, VSS (GND), DATA (RB7), and CLOCK (RB6) from the PIC programmer hardware through a properly prepared cable and it connects them to a 4PDT (four pole double throw) switch. There are two positions of the 4PDT switch. As seen from Figure 1.1, in the PROG position of the 4PDT switch, PIC16F1847 is ready to be programmed and in the RUN position, the loaded program is run. For programming the PIC16F1847 properly by means of a PIC programmer and the 4PDT switch, it is also a necessity to *switch off* the power switch. The **CPU section** consists of the PIC16F1847 microcontroller. In the project reported in

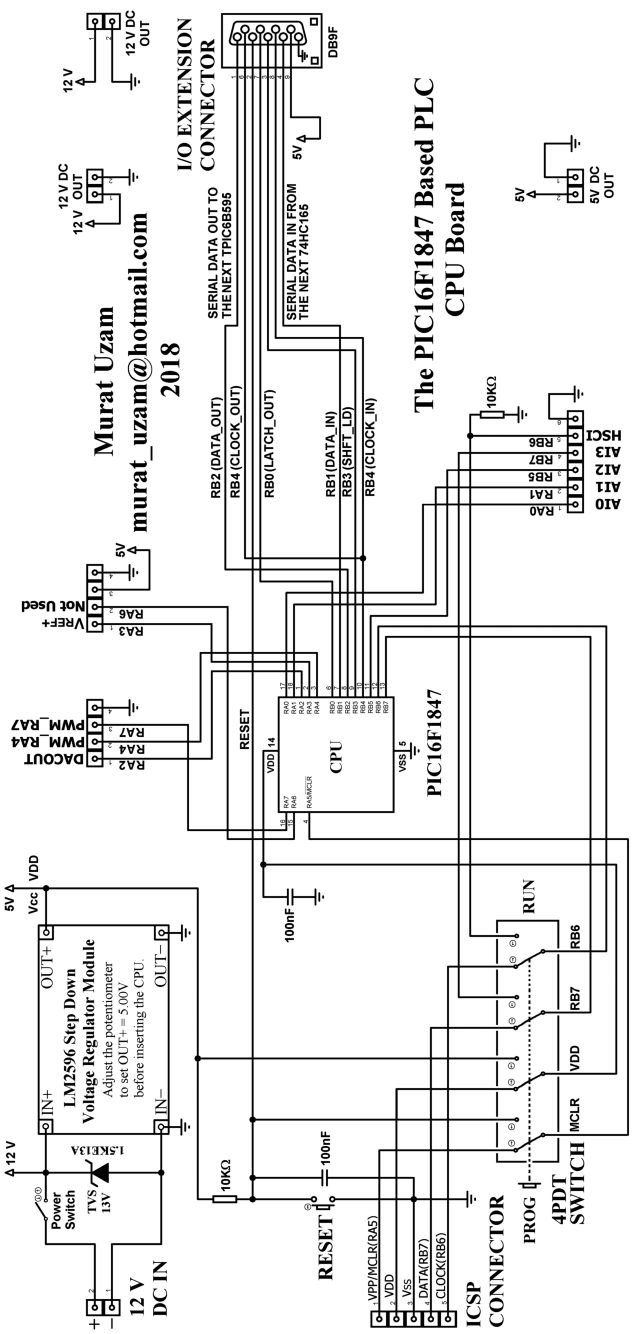


FIGURE 1.1 Schematic diagram of the PIC16F1847-Based PLC CPU board.

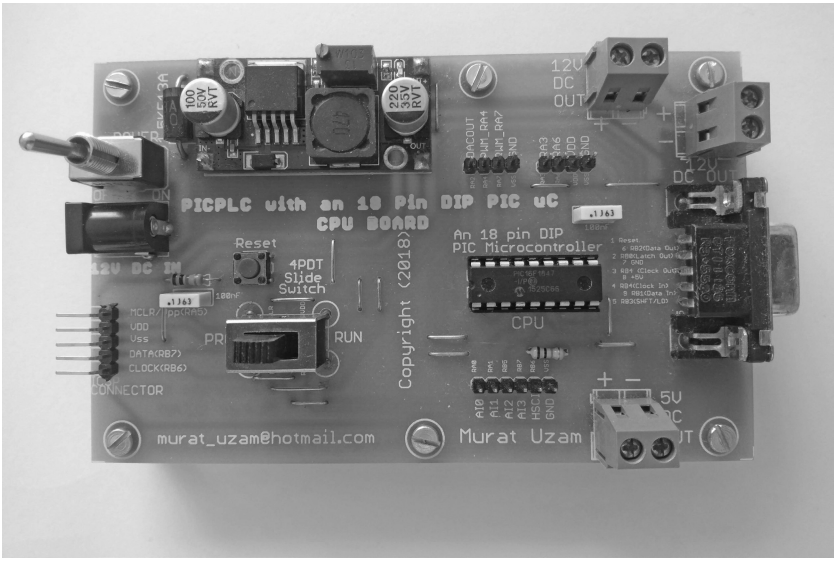


FIGURE 1.2 Photograph of the CPU board.

this book, the PLC is fixed to run at 32 MHz with an internal oscillator (oscillator frequency = 8 MHz and PLL = 4). This frequency is fixed because time delays are calculated based on this speed. RB1, RB3, and RB4 pins are all reserved to be used for 8-bit parallel-to-serial converter registers 74HC/LS165. Through these three pins and with added 74HC/LS165 registers we can describe as many inputs as necessary. RB1, RB3, and RB4 are the "data in", the "shift/load", and the "clock in" pins, respectively. Similarly, the RB2, RB4, and RB0 pins are all reserved to be used for 8-bit serial-to-parallel converter register/drivers TPIC6B595. Through these three pins and with added TPIC6B595 registers we can describe as many outputs as necessary. RB2, RB4, and RB0 are the "data out", the "clock out", and the "latch out" pins, respectively.

The RA0, RA1, RB5, and RB7 pins are described and used as analog inputs. They are called AI0, AI1, AI2, and AI3, respectively. The RA2 pin is used as an analog output and it is called DACOUT. The RA3 pin is used as VREF+ (ADC voltage reference input). The RB6 pin is used as the clock input of the high speed counter and it is called HSCI. The RA4 and RA7 pins are used as PWM (pulse width modulation) outputs. Therefore, they are called PWM_RA4 and PWM_RA7, respectively. The RA6 pin is not used. The PIC16F1847 provides the following—flash program memory (words): 8K; SRAM data memory (bytes): 1,024; and EEPROM data memory (bytes): 256. The PIC16F1847-Based PLC macros make use of registers defined in SRAM data memory.

Figures 1.3 and 1.4 show the schematic diagram and the photograph of the *I/O extension board*, respectively. The I/O extension board contains mainly *two sections*: 8 digital inputs and 8 digital outputs. The I/O extension connector DB9M, seen on the left, connects the I/O extension board to the CPU board or to a previous

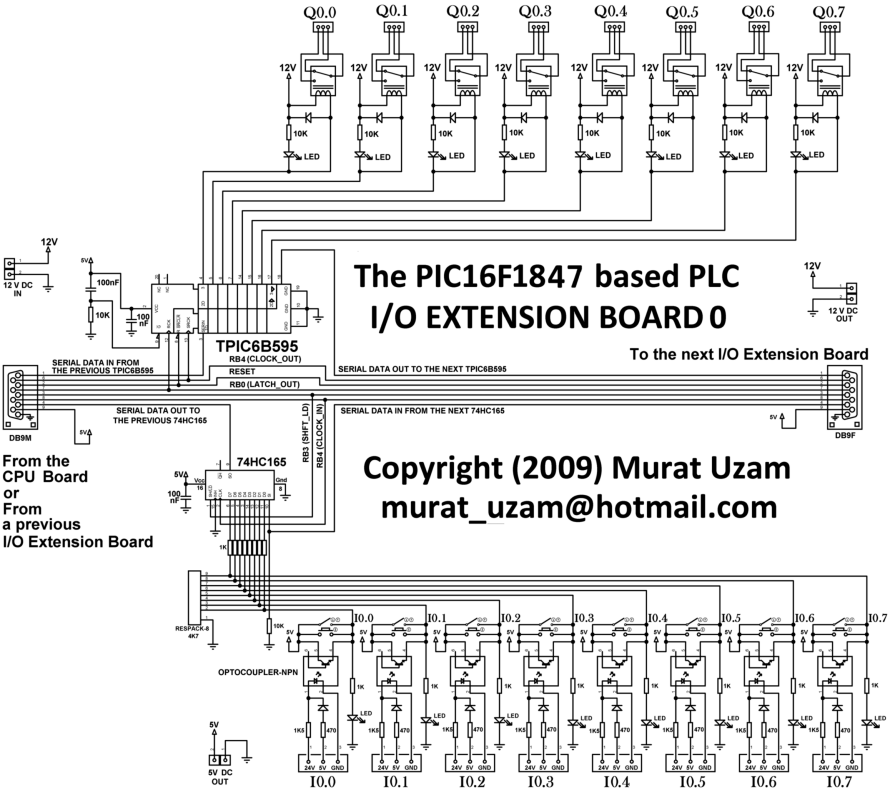


FIGURE 1.3 Schematic diagram of the I/O extension board.

I/O extension board. Similarly, the I/O extension connector DB9F, seen on the right, connects the I/O extension board to a next I/O extension board. In this way we can connect as many I/O extension boards as necessary. 5V DC and 12V DC are taken from the CPU board or from a previous I/O extension board and they are passed to the next I/O extension boards. All I/O data are sent to and taken from all the connected extension I/O boards by means of I/O extension connectors DB9M and DB9F.

The **inputs section** of each I/O extension board introduces 8 digital inputs for the PIC16F1847-Based PLC (called I0.0, I0.1, …, I0.7 for the first I/O extension board, called I1.0, I1.1, …, I1.7 for the second I/O extension board, called I2.0, I2.1, …, I2.7 for the third I/O extension board, and called I3.0, I3.1, …, I3.7 for the fourth and last I/O extension board). 5V DC or 24V DC input signals can be accepted by each input. These external input signals are isolated from the other parts of the hardware by using NPN-type optocouplers (e.g., 4N25). For simulating input signals, one can use on-board push buttons as temporary inputs and slide switches as permanent inputs. In the beginning of each PLC scan cycle (get_inputs), the 74HC/LS165 of each I/O extension board is loaded (RB3 [shift/load] = 0) with the level of 8 inputs, and then these data are serially clocked in (when RB3 = 1, through the RB1 "data in" and RB4

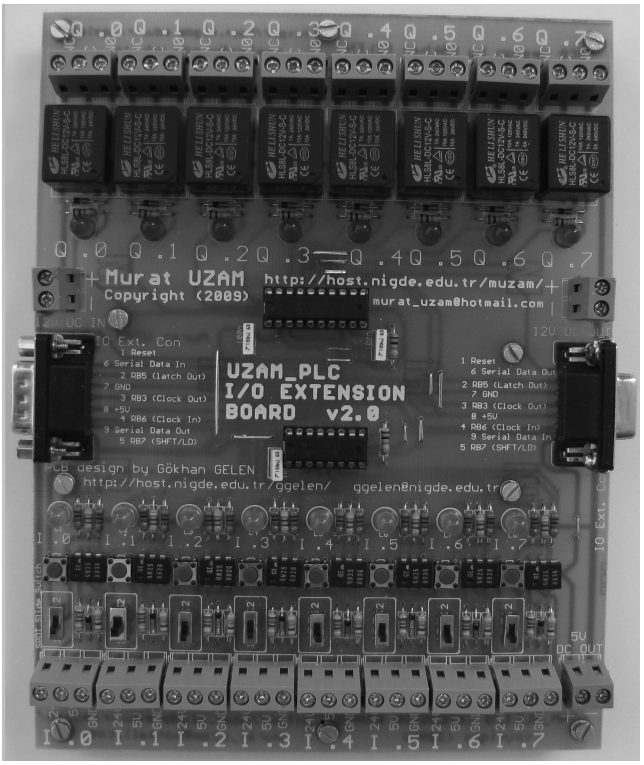


FIGURE 1.4 Photograph of the I/O extension board.

"clock in" pins). If there is only one I/O extension board used, then 8 clock_in signals are enough to get the 8 input signals. For each additional I/O extension board, 8 more clock_in signals are necessary. The serial data coming from the I/O extension board(s) are taken from the "SI" input of the 74HC/LS165.

The **outputs section** of each I/O extension board introduces 8 discrete relay outputs for the PIC16F1847-Based PLC (called Q0.0, Q0.1, …, Q0.7 for the first I/O extension board, called Q1.0, Q1.1, …, Q1.7 for the second I/O extension board, called Q2.0, Q2.1, …, Q0.7 for the third I/O extension board, and called Q3.0, Q3.1, …, Q3.7 for the fourth and last I/O extension board). Each relay operates with 12V DC and driven by an 8-bit serial-to-parallel converter register/driver TPIC6B595. Relays have SPDT (single pole double throw) contacts with C (common), NC (normally closed), and NO (normally open) terminals. At the end of each PLC scan cycle (send_outputs), the output data are serially clocked out (through the RB4 "clock out" and RB2 "data out" pins) and finally latched within the TPIC6B595. If there is only one I/O extension board used, then 8 clock_out signals are enough to send the 8 output signals. For each additional I/O extension board, 8 more clock_out signals are necessary. The serial data going to the I/O extension board(s) are sent out from the "SER OUT" (pin 18) of the TPIC6B595.

The PCB Gerber files of both the CPU board and the I/O extension board are downloadable from this book's webpage under the downloads section. Note that in the PCB design of the CPU board and the I/O extension board, some lines of I/O extension connectors DB9M and DB9F are different from the ones shown in Figures 1.1 and 1.3.

The project reported in this book makes use of a CPU board and four I/O extension boards. Thus, in total there are 32 digital inputs and 32 digital outputs. Figure 1.5 shows the PIC16F1847-Based PLC consisting of a CPU board, four I/O extension boards, a 12V DC adapter, and a PICkit 3 PIC programmer.

In addition to the CPU board and I/O extension boards, in this section let us briefly consider some additional input and output modules to be used with the PIC16F1847-Based PLC, as shown in Figure 1.6. The following is the list of these additional input and output modules:

1. Analog input modules
2. Analog output modules
3. RC low-pass filters module
4. 5.00V voltage reference module
5. Voltage regulator module

Analog input modules designed within this project are as follows:

1. 0V to 5V Analog Input Module 1
2. 0V to 5V Analog Input Module 2

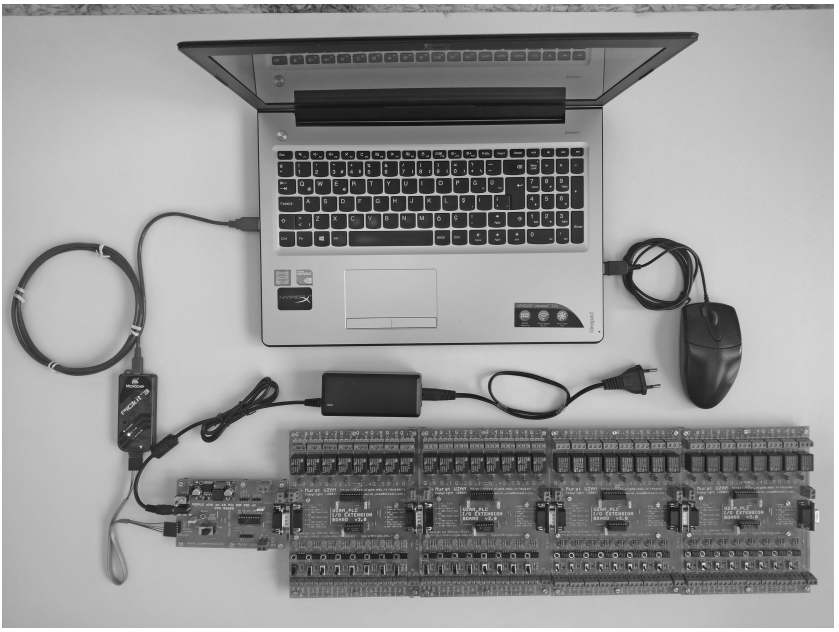


FIGURE 1.5 Photograph of the CPU board plus four I/O extension boards and a PICkit 3 PIC programmer.

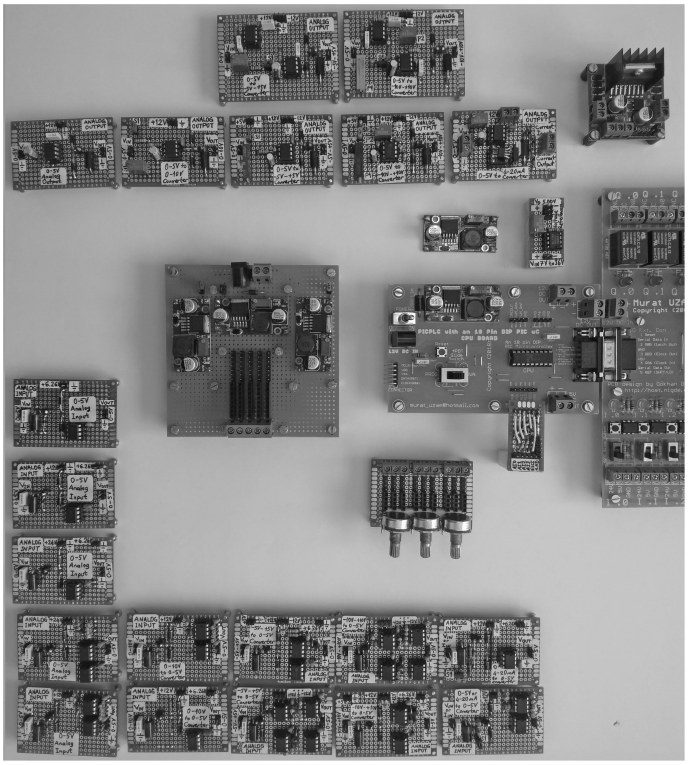


FIGURE 1.6 Photograph of the CPU board together with 13 analog input modules and 7 analog output modules.

3. 0V to 5V Analog Input Module 3
4. 0V to 5V Analog Input Module 4
5. 0V to 5V Analog Input Module 5
6. 0–10V to 0–5V Signal Converter—Analog Input Module 1
7. 0–10V to 0–5V Signal Converter—Analog Input Module 2
8. –5V – +5V to 0–5V Signal Converter—Analog Input Module 1
9. –5V – +5V to 0–5V Signal Converter—Analog Input Module 2
10. –10V – +10V to 0–5V Signal Converter—Analog Input Module 1
11. –10V – +10V to 0–5V Signal Converter—Analog Input Module 2
12. 0–5V or 4–20mA to 0–5V Signal Converter—Analog Input Module 1
13. 0–5V or 4–20mA to 0–5V Signal Converter—Analog Input Module 2

Analog output modules designed within this project are as follows:

1. 0V to 5V Analog Output Module
2. 0–5V to 0–10V Signal Converter—Analog Output Module
3. 0–5V to –5V – +5V Signal Converter—Analog Output Module 1
4. 0–5V to –5V – +5V Signal Converter—Analog Output Module 2

5. 0–5V to –10V – +10V Signal Converter—Analog Output Module 1
6. 0–5V to –10V – +10V Signal Converter—Analog Output Module 2
7. 0–5V to 4–20mA Signal Converter—Analog Output Module

These analog input and analog output modules are explained in detail in Chapter 6 of the *Advanced Concepts* book.

An RC low-pass filter is a filter circuit, composed of a resistor and a capacitor, which passes low-frequency signals and blocks high-frequency signals. When a resistor is placed in series with the power source and a capacitor is placed parallel to that same power source, this type of circuit forms a low-pass filter. Figure 1.7 depicts the schematic diagram of RC low-pass filters constructed for analog inputs AI0, AI1, AI2, and AI3, with the cut-off frequency of 48Hz.

An external 5.00V voltage reference is necessary to be used with the analog-to-digital converter (ADC) module and the digital-to-analog converter (DAC) module of the PIC16F1847. To satisfy this requirement, a low-cost solution is obtained by using the REF02 voltage reference from Analog Devices. Figure 1.8(a) shows the schematic diagram of the 5.00V voltage reference REF02 with a trim adjustment circuit consisting of R1, R2, and POT, while Figure 1.8(b) depicts the photograph of the 5.00V voltage reference module.

In analog input modules and analog output modules (see Chapter 6 of the *Advanced Concepts* book) +5.00V and +6.26V power supplies are necessary, and in the DC motor control examples with an L298N dual full-bridge driver (see "Application Examples"), a +6.00V power supply is necessary. As considered before, LM2596 step-down voltage regulators can be used to obtain these DC voltages from the 12V DC input voltage. To address this need, a voltage regulator module is designed. Figure 1.9(a) shows the schematic diagram of the voltage regulator module, consisting of three LM2596 step-down voltage regulators, while Figure 1.9(b) depicts the photograph of the voltage regulator module. By using this voltage regulator module,

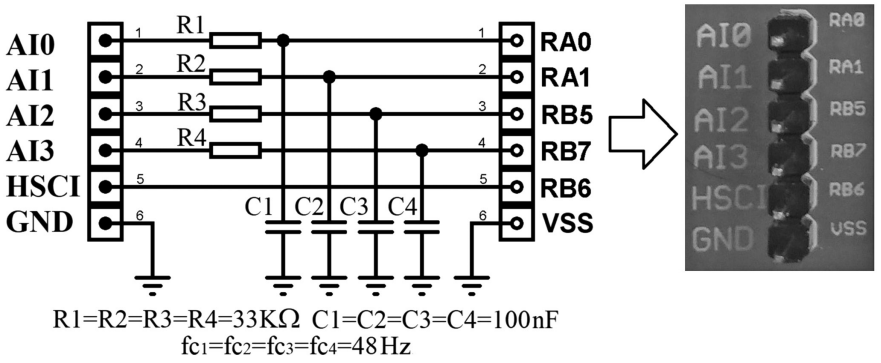


FIGURE 1.7 Schematic diagram of RC low-pass filters for analog inputs AI0, AI1, AI2, and AI3.

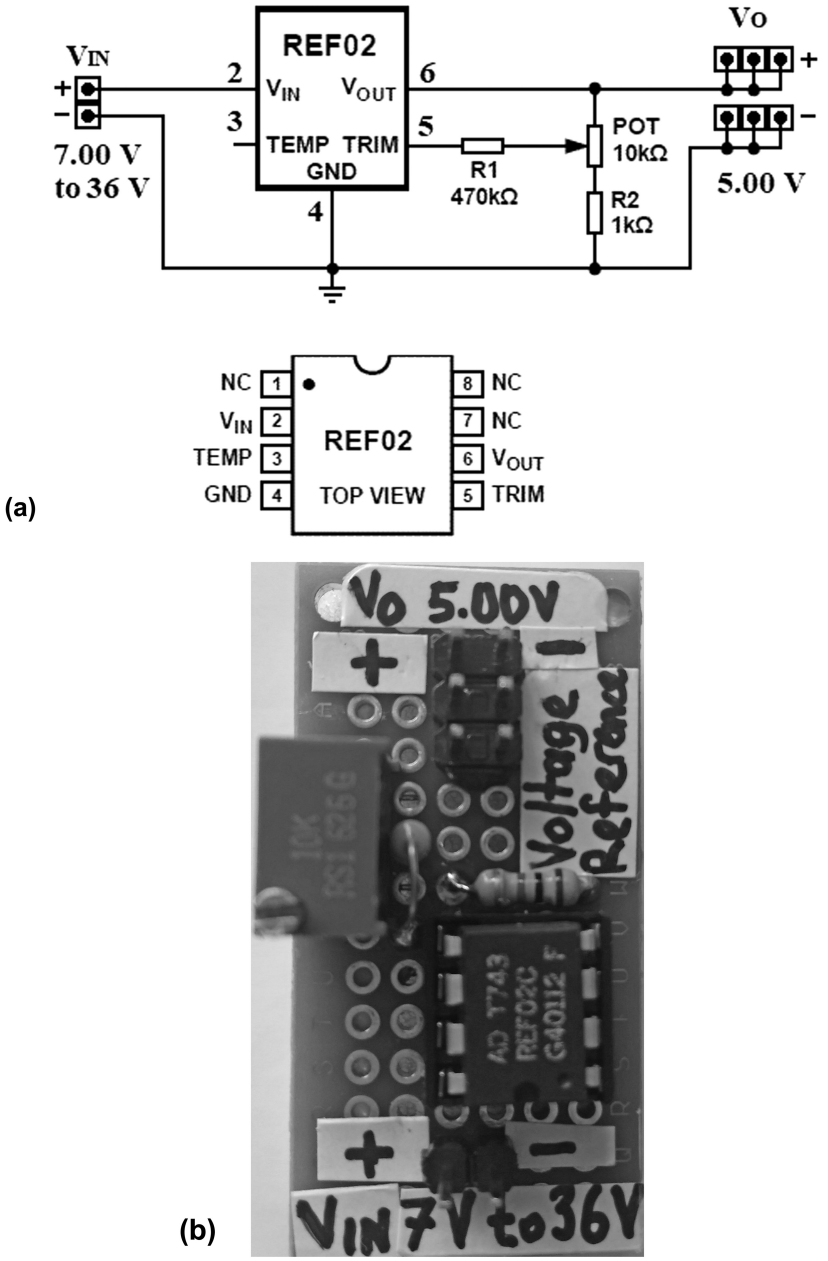


FIGURE 1.8 (a) Schematic diagram of the 5.00V voltage reference REF02 with a trim adjustment circuit consisting of R1, R2, and POT; (b) Photograph of the 5.00V voltage reference module.

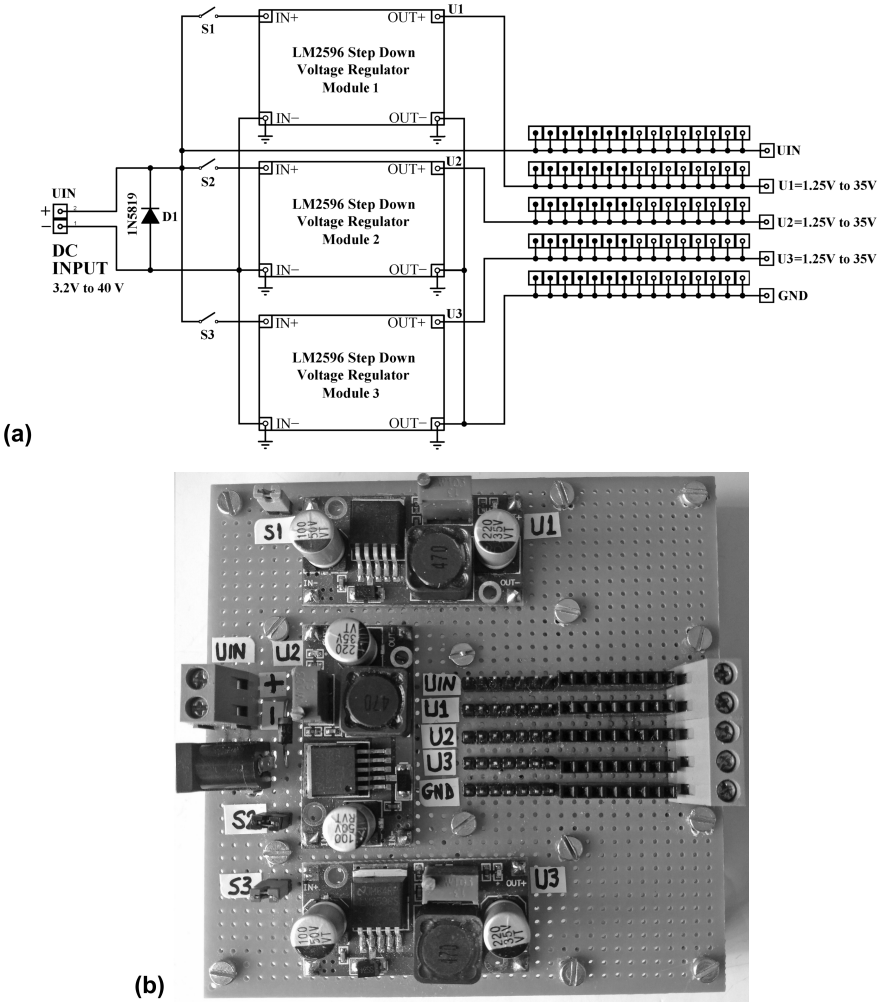


FIGURE 1.9 (a) Schematic diagram of the voltage regulator module, consisting of three LM2596 step-down voltage regulators; (b) Photograph of the voltage regulator module.

three independent voltage values can be adjusted and used. D1 is used to make sure that the polarity of the DC input voltage is correct. Switches S1, S2, and S3 (implemented by using jumpers) are used to turn on or off the LM2596S voltage regulators 1, 2, and 3, respectively.

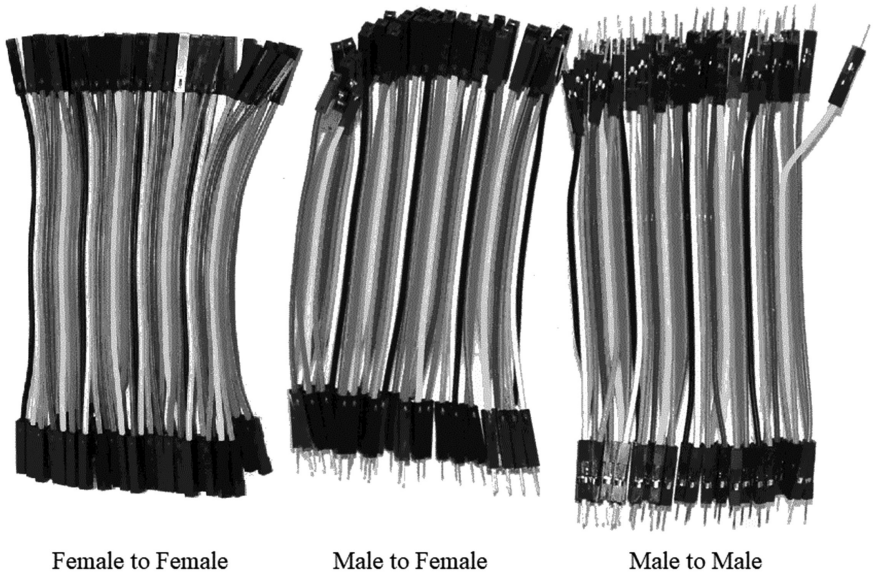


FIGURE 1.10 Three types of Dupont cables used in the project described in this book.

Last but not least, in order to connect the above-mentioned input and output modules with the PIC16F1847-Based PLC input/output terminals, it is necessary to use some cables. For this purpose, three types of Dupont cables, shown in Figure 1.10, are used.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

2 Basic Software

INTRODUCTION

In this chapter, the basic software of the PIC16F1847-Based PLC is explained. A PLC scan cycle includes the following: obtain the inputs, run the user program, and update the outputs. It is also necessary to define and initialize all variables used within a PLC. Necessary functions are all described as PIC Assembly macros to be used in the PIC16F1847-Based PLC. As can be seen from Figure 2.1, the source files and their macros developed in the PICPLC_PIC16F1847 project file are as follows:

1. PICPLC_PIC16F1847_memory.inc
2. PICPLC_PIC16F1847_main.asm
3. PICPLC_PIC16F1847_user_Bsc.inc
4. PICPLC_PIC16F1847_subr.inc
5. PICPLC_PIC16F1847_macros_Bsc.inc
   - 5.1 initialize (for PLC initialization)
   - 5.2 ISR (interrupt service routines)
   - 5.3 get_inputs (for handling the inputs)
   - 5.4 lpf_progs (low-pass digital filter macros for analog inputs)
   - 5.5 send_outputs (for sending the outputs)

The basic software of the PIC16F1847-Based PLC makes use of general-purpose 8-bit registers (GPR) of SRAM data memory of the PIC16F1847 microcontroller. 1,024 SRAM bytes of PIC16F1847 are allocated in 13 banks, namely Bank0, Bank1, …, Bank12. In this PLC project, 695 SRAM bytes are defined and reserved to be used within the PLC functions. GPRs in banks Bank0, Bank1, Bank2, and Bank3 are intentionally left unused for general use. Thus there are 329 GPRs ready to be used. The directory called "PICPLC_PIC16F1847_Bsc", downloadable from this book's webpage under the downloads section, contains all project files, macros, definitions, and examples necessary for the PIC16F1847-Based PLC project explained in this book (*Hardware and Basic Concepts*).

Note that files "PICPLC_PIC16F1847_macros_Bsc.inc" and "PICPLC_PIC16F1847_user_Bsc.inc" refer to the macros and user program files of the basic concepts developed in the PIC16F1847-Based PLC project, respectively. They do not contain files related to the intermediate and advanced concepts. These files are intended for the readers who purchased this book as a standalone book. On the other hand, when this book is purchased as a part of the set of three books, all project files including basic, intermediate, and advanced concepts are put in the same directory and the reader is entitled to download and use the whole of the project files in one directory, the name of which becomes "PICPLC_PIC16F1847" instead of "PICPLC_PIC16F1847_Bsc". Therefore, in the second case, the name of the file

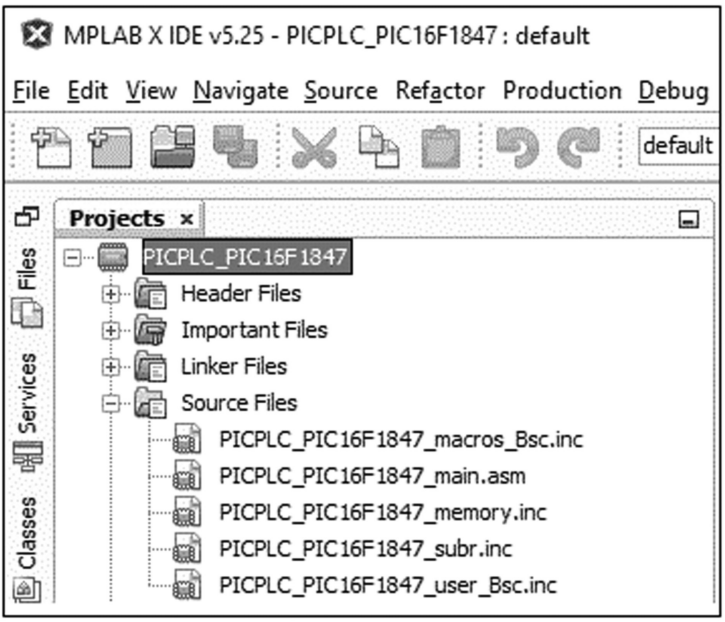


FIGURE 2.1 Screenshot of the "PICPLC_PIC16F1847" project, showing the five source files developed and used in the project.

"PICPLC_PIC16F1847_macros_Bsc.inc" (and PICPLC_PIC16F1847_user_Bsc .inc, respectively) becomes "PICPLC_PIC16F1847_macros.inc" (and PICPLC_ PIC16F1847_user.inc, respectively).

In this section the contents of the source files depicted in Figure 2.1 are explained. In addition, the concept of a "contact bouncing" problem and how it is solved in the PIC16F1847-Based PLC are explained in detail.

2.1 DEFINITION AND ALLOCATION OF VARIABLES

The definitions of all 8-bit variables to be used for the PIC16F1847-Based PLC project and their allocation in SRAM data memory are shown in Figures 2.2 and 2.3, respectively. These definitions are placed in the "PICPLC_PIC16F1847_macros _Bsc.inc" file. Although detailed explanations for these variables are provided in the related sections of this book, let us now briefly consider these 8-bit variables. In this project, we define four 8-bit registers (I0, I1, I2, and I3) to hold the debounced state of physical digital input registers (74HC/LS165) and four 8-bit registers (Q0, Q1, Q2, and Q3) to hold the state of physical digital output registers. Temp_1 and Temp_2 are general temporary registers declared to be used in some macros. SMB1 is declared to be used for obtaining special memory bits. SMB2 is declared to be used for obtaining reference timing signals.

It is well known that digital inputs taken from contacts always suffer from "contact bouncing". To circumvent this problem, we define a "debouncing" mechanism for the digital inputs, and this will be explained later. In the "get_inputs" stage of the

```
;-------------------------------------------------------------------------------;
;       VARIABLE DEFINITIONS                                                    ;
;-------------------------------------------------------------------------------;
;-------------------- beginning of BANK0 ---------------------------------------
        cblock 0x020    ; There are 80 8-bit GPRs available in BANK0.

        endc            ;
;-------------------- end of BANK0 ---------------------------------------------
;-------------------- beginning of common RAM memory ---------------------------
        cblock 0x70
        I0, I1, I2, I3, Q0, Q1, Q2, Q3, Temp_1, Temp_2, SMB1, SMB2
        endc
;-------------------- end of common RAM memory ----------------------------------
;
;-------------------- beginning of BANK1 ---------------------------------------
        cblock 0x0A0    ; There are 80 8-bit GPRs available in BANK1.

        endc            ;
;-------------------- end of BANK1 ---------------------------------------------
;-------------------- beginning of BANK2 ---------------------------------------
        cblock 0x120    ; There are 80 8-bit GPRs available in BANK2.

        endc            ;
;-------------------- end of BANK2 ---------------------------------------------
;-------------------- beginning of BANK3 ---------------------------------------
        cblock 0x1A0    ;There are 80 8-bit GPRs available in BANK3.

        endc            ;
;-------------------- end of BANK3 ---------------------------------------------
;-------------------- beginning of BANK4 ---------------------------------------
        cblock 0x220    ; 80 8-bit-variables are defined to hold
        TV_L            ; low byte timing values
        endc            ; TV_L, TV_L+1, ..., TV_L+79
;-------------------- end of BANK4 ---------------------------------------------
;-------------------- beginning of BANK5 ---------------------------------------
        cblock 0x2A0    ; 80 8-bit-variables are defined to hold
        TV_H            ; high byte timing values
        endc            ; TV_H, TV_H+1, ..., TV_H+79
;-------------------- end of BANK5 ---------------------------------------------
;
;-------------------- beginning of BANK6 ---------------------------------------
        cblock 0x320    ; 80 8-bit-variables are defined to hold
        CV_L            ; low byte count values
        endc            ; CV_L, CV_L+1, ..., CV_L+79
;-------------------- end of BANK6 ---------------------------------------------
;
;-------------------- beginning of BANK7 ---------------------------------------
        cblock 0x3A0    ; 80 8-bit-variables are defined to hold
        CV_H            ; high byte count values
        endc            ; CV_H, CV_H+1, ..., CV_H+79
;-------------------- end of BANK7 ---------------------------------------------
```

FIGURE 2.2 (*1 of 5*) Definition of 8-bit variables.

PLC scan cycle, digital input signals are serially taken from the related 74HC/LS165 registers and stored in the SRAM registers. As a result, bI0, bI1, bI2, and bI3 will hold these bouncing digital input signals. After applying the debouncing mechanism to the bouncing digital input signals bI0, bI1, bI2, and bI3, we obtain "debounced" input signals and they are stored in SRAM registers I0, I1, I2, and I3 respectively. In the "send_outputs" stage of the PLC scan cycle, the output information stored in

```
;---------------------- beginning of BANK8 ---------------------------------------------
        cblock 0x420    ; 8 Memory bytes, 8x8=64 Memory bits (Internal Relays)
        M0, M1, M2, M3, M4, M5, M6, M7
        endc            ;
        cblock 0x428    ; 8 Memory bytes, 8x8=64 Memory bits (Internal Relays)
        M8, M9, M10, M11, M12, M13, M14, M15
        endc            ;
        cblock 0x430    ; 8 Memory bytes, 8x8=64 Memory bits (Internal Relays)
        M16, M17, M18, M19, M20, M21, M22, M23
        endc            ;
        cblock 0x438    ; 8 Memory bytes, 8x8=64 Memory bits (Internal Relays)
        M24, M25, M26, M27, M28, M29, M30, M31
        endc            ;
        cblock 0x440    ; 8 Memory bytes, 8x8=64 Memory bits (Internal Relays)
        M32, M33, M34, M35, M36, M37, M38, M39
        endc            ;
        cblock 0x448    ; 8 Memory bytes, 8x8=64 Memory bits (Internal Relays)
        M40, M41, M42, M43, M44, M45, M46, M47
        endc            ;
        cblock 0x450    ; 8 Memory bytes, 8x8=64 Memory bits (Internal Relays)
        M48, M49, M50, M51, M52, M53, M54, M55
        endc            ;
        cblock 0x458    ; 8 Memory bytes, 8x8=64 Memory bits (Internal Relays)
        M56, M57, M58, M59, M60, M61, M62, M63
        endc            ;
        cblock 0x460    ; 8 Memory bytes, 8x8=64 Memory bits (Internal Relays)
        M64, M65, M66, M67, M68, M69, M70, M71
        endc            ;
        cblock 0x468    ; 8 Memory bytes, 8x8=64 Memory bits (Internal Relays)
        M72, M73, M74, M75, M76, M77, M78, M79
        endc            ; In BANK8 80 Memory bytes (640 Memory bits) are defined
;---------------------- end of BANK8 -----------------------------------------------------
```

FIGURE 2.2 Continued

the 8-bit SRAM registers Q0, Q1, Q2, and Q3 is serially sent out to and stored in the related TPIC6B595 registers. This means that the Q0, Q1, Q2, and Q3 registers will hold output information and their contents will be copied into the TPIC6B595 registers at the end of each PLC scan cycle.

160 8-bit registers, namely TV_L, TV_L+1, …, TV_L+79 and TV_H, TV_H+1, …, TV_H+79, are defined to be used in timer macros (see Chapter 5 of this book) for holding current timing values of timers. Ten 8-bit registers, namely T_Q0, T_Q1, …, T_Q9 are defined to be used in timer macros for holding timer status bits (timer outputs). 160 8-bit registers, namely CV_L, CV_L+1, …, CV_L+79 and CV_H, CV_H+1, …, CV_H+79, are defined to be used in counter macros (see Chapter 6 of this book) for holding current count values of counters. 20 8-bit registers, namely C_Q0, C_Q1, …, C_Q9 and C_QD0, C_QD1, …, C_QD9, are defined to be used in counter macros for holding counter status bits (counter outputs). 128 8-bit registers, namely M0, M1, …, M127, are defined for obtaining 1,024 memory bits (internal relays, in PLC jargon). The following 43 8-bit registers are defined to be used in drum sequencer instruction: drum_TVL, drum_TVL+1, …, drum_TVL+15, drum_TVH, drum_TVH+1, …, drum_TVH+15, drum_TQL, drum_TQH, drum_stepsL, drum_stepsH, drum_eventsL, drum_eventsH, drum_QL, drum_QH, drum_tmp, drum_tmpL, and drum_tmpH. The following 54 8-bit registers are defined to be

```
        ;
        ;-------------------- beginning of BANK9 ---------------------------------------------
        cblock 0x4A0    ; 8 Memory bytes,  8x8=64 Memory bits (Internal Relays)
        M80, M81, M82, M83, M84, M85, M86, M87
        endc            ;
        cblock 0x4A8    ; 8 Memory bytes,  8x8=64 Memory bits (Internal Relays)
        M88, M89, M90, M91, M92, M93, M94, M95
        endc            ;
        cblock 0x4B0    ; 8 Memory bytes,  8x8=64 Memory bits (Internal Relays)
        M96, M97, M98, M99, M100, M101, M102, M103
        endc            ;
        cblock 0x4B8    ; 8 Memory bytes,  8x8=64 Memory bits (Internal Relays)
        M104, M105, M106, M107, M108, M109, M110, M111
        endc            ;
        cblock 0x4C0    ; 8 Memory bytes,  8x8=64 Memory bits (Internal Relays)
        M112, M113, M114, M115, M116, M117, M118, M119
        endc            ;
        cblock 0x4C8    ; 8 Memory bytes,  8x8=64 Memory bits (Internal Relays)
        M120, M121, M122, M123, M124, M125, M126, M127
        endc            ; In BANK9 48 Memory bytes (384 Memory bits) are defined.
;-----------------------------------------------------------------
        ; In BANK8 and BANK9, 128 Memory bytes (1024 Memory bits) are defined.
        ; M0, M1, ..., M127
;-----------------------------------------------------------------
        cblock 0x4D0    ; Timer status registers
        T_Q0, T_Q1, T_Q2, T_Q3, T_Q4, T_Q5, T_Q6, T_Q7, T_Q8, T_Q9
        endc
        cblock 0x4DA    ; Counter status registers:
        C_Q0, C_Q1, C_Q2, C_Q3, C_Q4, C_Q5, C_Q6, C_Q7, C_Q8, C_Q9
        endc
        cblock 0x4E4    ; Down Counter status registers:
        C_QD0,C_QD1,C_QD2,C_QD3,C_QD4,C_QD5,C_QD6,C_QD7,C_QD8,C_QD9
        endc            ;
        cblock 0x4EE    ;
        ; --------- 2 RAM locations in BANK9: 4EEh & 4EFh are not used
        endc            ;
;-------------------- end of BANK9 ----------------------------------------------
```

FIGURE 2.2 Continued

used in SFC (sequential function charts)-related macros (see Chapter 5 of *Advanced Concepts*): step_1.TL, step_1.TL+1, …, step_1.TL+24, step_1.TH, step_1.TH+1, …, step_1.TH+24, SF0, SF1, SF2, MB0, MB1, and MB2. 40 8-bit registers, namely LPF, LPF+1, …, LPF+39, are defined to be used in low-pass digital filter macros for holding current timing values of low-pass digital filters. The following eight 8-bit registers hold four 10-bit noisy digital values for 4 analog inputs: nAI0L, nAI0H, nAI1L, nAI1H, nAI2L, nAI2H, nAI3L, and nAI3H. The following eight 8-bit registers hold four 10-bit filtered digital values for 4 analog inputs: AI0L, AI0H, AI1L, AI1H, AI2L, AI2H, AI3L, and AI3H. Registers HSC_B2 and HSC_B3 are defined to be used in the HSC_RB6 macro (see Chapter 2 of *Advanced Concepts*) to hold the most significant two bytes of 32-bit count values. 32 8-bit registers, namely DBNCR, DBNCR+1, …, DBNCR+31, are defined to be used in the debouncer macro "dbncrN" for holding current timing values of debouncer macros. 8-bit registers CNT1, CNT2, and CNT3 are defined to be used in the "ISR" macro in order to obtain reference timing signals T_2ms, T_10ms, T_100ms, and T_1s. 8-bit registers TenK, Thou, Hund, Tens, and Ones are defined to be used in the following macros: "Conv_UInt_2_BCD_P",

```
        ;-------------------- beginning of BANK10 -------------------------------------------
        cblock 0x520    ; 16 8-bit-variables are defined for d_TON16
        drum_TVL        ; to hold low byte timing values
        endc            ; drum_TVL, drum_TVL +1, ..., drum_TVL +15
        cblock 0x530    ; 16 8-bit-variables are defined for d_TON16
        drum_TVH        ; to hold high byte timing values
        endc            ; drum_TVH, drum_TVH +1, ..., drum_TVH +15
        cblock 0x540    ;16 Status bits for 16 d_TON16
        drum_TQL,drum_TQH
        endc
        cblock 0x542    ;16 Steps for Drum Sequencer Instruction
        drum_stepsL,drum_stepsH
        endc
        cblock 0x544    ;16 drum events for Drum Sequencer Instruction
        drum_eventsL,drum_eventsH
        endc
        cblock 0x546
        drum_QL,drum_QH;16 final drum outputs
        endc
        cblock 0x548    ;These 3 registers are used in Drum Sequencer Instruction.
        drum_tmp,drum_tmpL,drum_tmpH
        endc
        cblock 0x54B    ;
        SF0,SF1,SF2     ;24 step flags defined for SFC
        endc            ;
        cblock 0x54E    ;
        MB0,MB1,MB2 ;24 Memory bits defined for SFC
        endc            ;
        cblock 0x551    ;24 Memory words defined for SFC to be used in elapsed times
        step_1.TL,step_1.TH,step_2.TL,step_2.TH,step_3.TL,step_3.TH,step_4.TL,step_4.TH
        endc            ;
        cblock 0x559    ;24 Memory words defined for SFC to be used in elapsed times
        step_5.TL,step_5.TH,step_6.TL,step_6.TH,step_7.TL,step_7.TH,step_8.TL,step_8.TH
        endc            ;
        cblock 0x561    ;24 Memory words defined for SFC to be used in elapsed times
        step_9.TL,step_9.TH,step_10.TL,step_10.TH,step_11.TL,step_11.TH,step_12.TL,step_12.TH
        endc            ;
        cblock 0x569    ;24 Memory words defined for SFC to be used in elapsed times
        step_13.TL,step_13.TH,step_14.TL,step_14.TH,step_15.TL,step_15.TH,step_16.TL
        endc            ;
;-------------------------------------------------------------------
;-------------------- end of BANK10 ------------------------------------------
;
```

FIGURE 2.2 Continued

"Conv_BCD_U_2_Uint", "Conv_BCD_P_2_Uint", "Conv_UsInt_2_BCD_U", and "Conv_UsInt_2_BCD_P". The 8-bit register "STP_bits" is defined to be used in the PWM macros and the HSC macro. 8-bit registers i, j, and k are defined to be used in the selection macros (see Chapter 2 of *Advanced Concepts*).

The individual bits (1-bit variables) of 8-bit SRAM registers M0, M1, M2, …, M127 are all considered in the next section. The definitions of 1-bit (Boolean) variables are placed in the "PICPLC_PIC16F1847_macros_Bsc.inc" file. The definitions of 32 bouncing digital input signals bI0.0, bI0.1, …, bI3.7 by using all bits of 8-bit SRAM registers bI0, bI1, bI2, and bI3 are shown in Figure 2.4.

The allocation of individual bits (1-bit variables) of 8-bit SRAM registers bI0, bI1, bI2, and bI3 is shown in Table 2.1.

```
;---------------------- beginning of BANK11 ---------------------------------------------
;---------------------- LPF Variables are in BANK11 -------------------------------
        cblock 0x5A0    ; 40 8-bit-variables are defined for low pass digital filters
        LPF             ; LPF, LPF+1, ..., LPF+39
        endc            ;
        cblock 0x5C8    ; 4 noisy Digital Values for 4 Analog inputs
        nAI0L, nAI0H, nAI1L, nAI1H, nAI2L, nAI2H, nAI3L, nAI3H
        endc            ; are stored in these registers
        cblock 0x5D0    ; Filtered Digital Values for 4 Analog inputs
        AI0L, AI0H, AI1L, AI1H, AI2L, AI2H, AI3L, AI3H
        endc            ; are stored in these registers
        cblock 0x5D8    ;
        step_16.TH      ;
        endc            ;
        cblock 0x5D9    ;24 Memory words defined for SFC to be used in elapsed times
        step_17.TL,step_17.TH,step_18.TL,step_18.TH,step_19.TL,step_19.TH,step_20.TL,step_20.TH
        endc            ;
        cblock 0x5E1    ;24 Memory words defined for SFC to be used in elapsed times
        step_21.TL,step_21.TH,step_22.TL,step_22.TH,step_23.TL,step_23.TH,step_24.TL,step_24.TH
        endc            ;
        cblock 0x5E9    ; HSC_B2 and HSC_B3 registers are used in the HSC_RB6 macro.
        HSC_B2,HSC_B3
        endc            ;
        cblock 0x5EB    ;
                        ;5 Bytes are available.
        endc            ;
;---------------------- end of BANK11 ---------------------------------------------
;
;---------------------- beginning of BANK12 ---------------------------------------------
;---------------------- Debouncer Variables are in BANK12 -------------------------
        cblock 0x620            ; 32 8-bit-variables are defined to hold timing values
        DBNCR                   ; DBNCR, DBNCR+1, ..., DBNCR+31
        endc                    ;
        cblock 0x640            ; 32 bouncing digital inputs are stored in these four registers
        bI0, bI1, bI2, bI3      ; bI0.0, bI0.1, ..., bI3.7
        endc                    ;
        cblock 0x644            ;
        CNT1, CNT2, CNT3        ;These 3 registers are used in the ISR.
        endc                    ;
        cblock 0x647            ;These five temporary registers are defined for
        TenK,Thou,Hund,Tens,Ones;"Conv_UInt_2_BCD_P", "Conv_BCD_U_2_UInt"
        endc                    ;"Conv_BCD_P_2_UInt", "Conv_UsInt_2_BCD_U",
                                ;and "Conv_UsInt_2_BCD_P" macros.
        cblock 0x64C            ;
        STP_bits                ;This setup register is used in
        endc                    ;PWM macros and HSC_RB6 macro.
        cblock 0x64D            ;
        i,j,k                   ;These registers are used in the selection macros.
        endc                    ;
;---------------------- end of BANK12 ---------------------------------------------
```

FIGURE 2.2 Continued

The definitions of 32 debounced digital input signals I0.0, I0.1, …, I3.7 by using all bits of 8-bit SRAM registers I0, I1, I2, and I3 are shown in Figure 2.5.

The allocation of individual bits (1-bit variables) of 8-bit SRAM registers I0, I1, I2, and I3 is shown in Table 2.2.

The definitions of 32 digital output signals Q0.0, Q0.1, …, Q3.7 by using all bits of 8-bit SRAM registers Q0, Q1, Q2, and Q3 are shown in Figure 2.6.

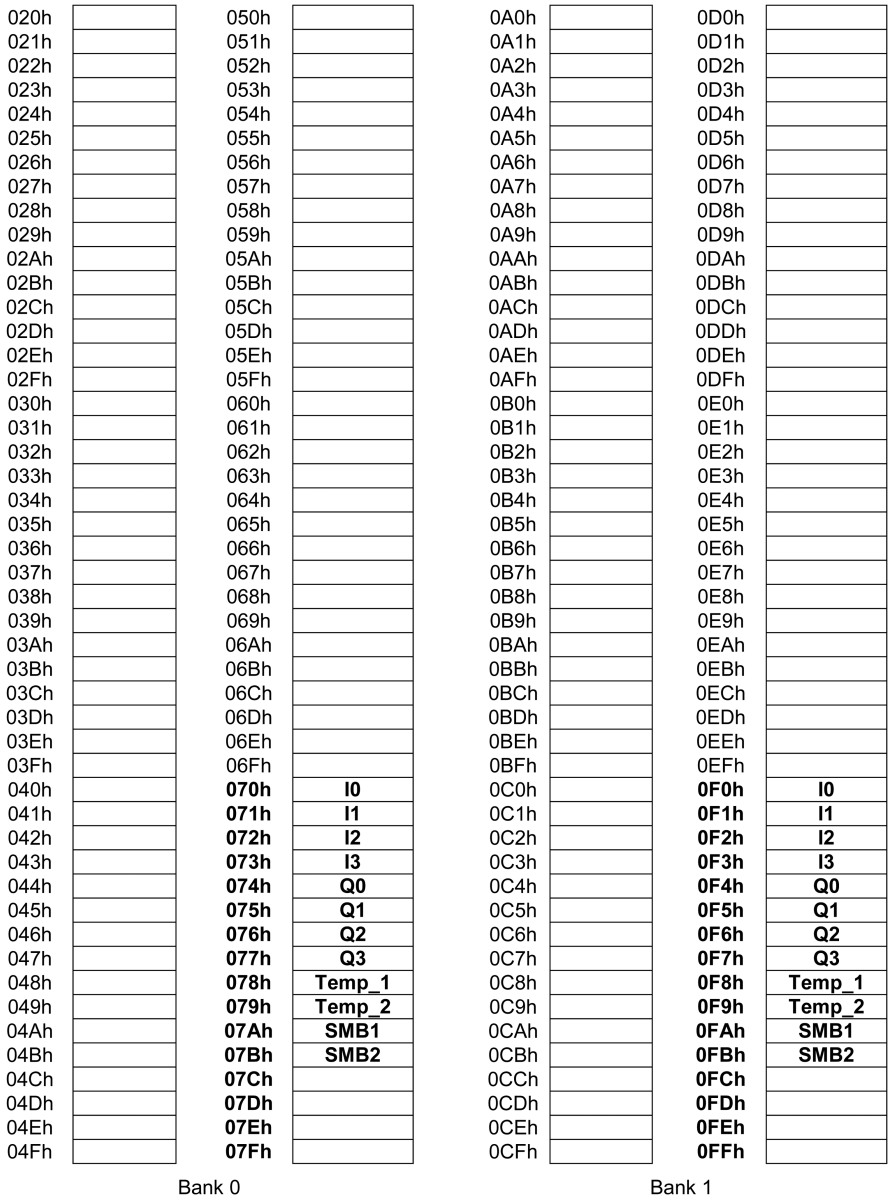


FIGURE 2.3 (*1 of 7*) Allocation of 8-bit variables in SRAM data memory.

The allocation of individual bits (1-bit variables) of 8-bit SRAM registers Q0, Q1, Q2, and Q3 is shown in Table 2.3.

The definitions of special memory bits and for 74HC165 and TPIC6B595 ICs are depicted in Figure 2.7(a) and (b), respectively. Tables 2.4 and 2.5 show the allocation of individual bits of the SMB1 register and SMB2 register, respectively.

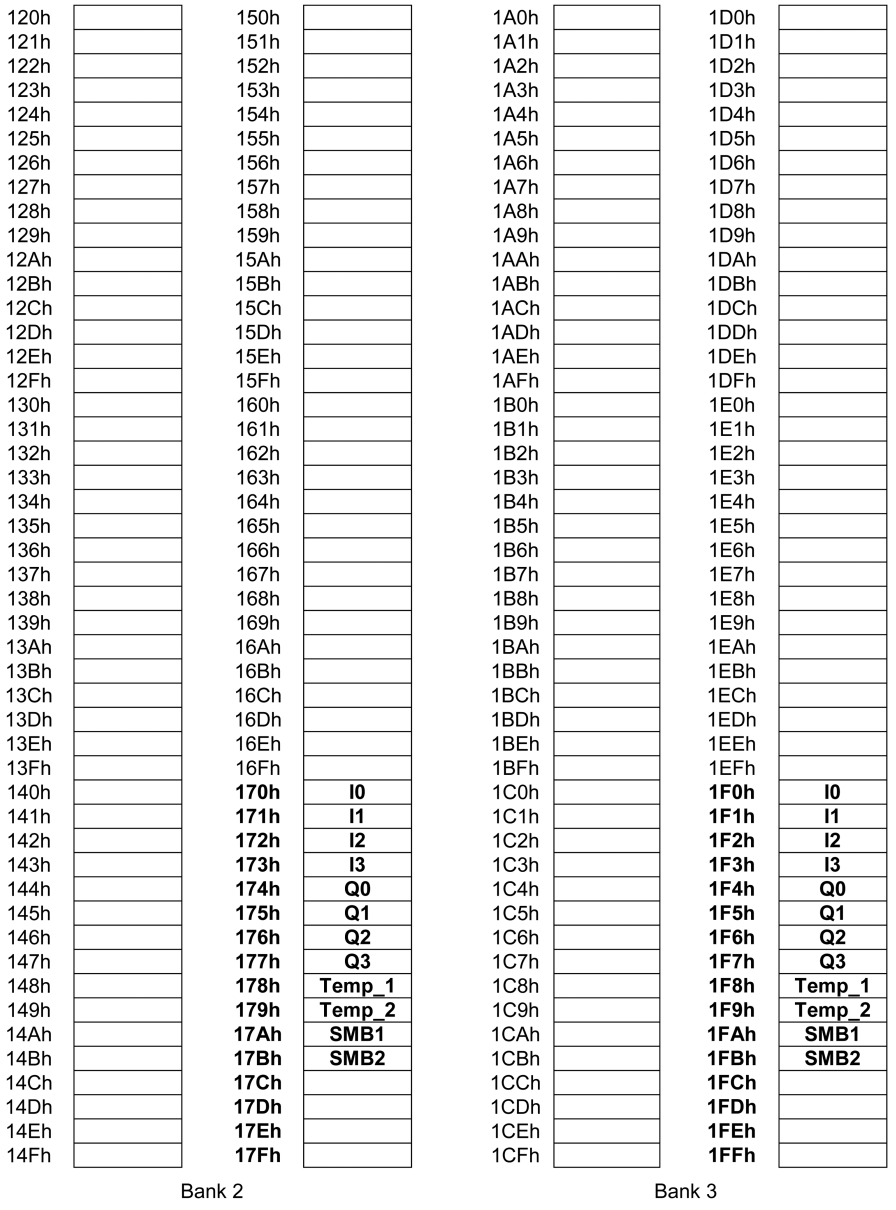


FIGURE 2.3 Continued

The variable "LOGIC0" is defined to hold a logic "0" value throughout the PLC operation. At the initialization stage it is deposited with this value. Similarly, the variable "LOGIC1" is defined to hold a logic "1" value throughout the PLC operation. At the initialization stage it is deposited with this value. The special memory bit "FRSTSCN" is arranged to hold the value of "1" at the first PLC scan cycle only.

220h	TV_L	250h	TV_L+48	2A0h	TV_H	2D0h	TV_H+48
221h	TV_L+1	251h	TV_L+49	2A1h	TV_H+1	2D1h	TV_H+49
222h	TV_L+2	252h	TV_L+50	2A2h	TV_H+2	2D2h	TV_H+50
223h	TV_L+3	253h	TV_L+51	2A3h	TV_H+3	2D3h	TV_H+51
224h	TV_L+4	254h	TV_L+52	2A4h	TV_H+4	2D4h	TV_H+52
225h	TV_L+5	255h	TV_L+53	2A5h	TV_H+5	2D5h	TV_H+53
226h	TV_L+6	256h	TV_L+54	2A6h	TV_H+6	2D6h	TV_H+54
227h	TV_L+7	257h	TV_L+55	2A7h	TV_H+7	2D7h	TV_H+55
228h	TV_L+8	258h	TV_L+56	2A8h	TV_H+8	2D8h	TV_H+56
229h	TV_L+9	259h	TV_L+57	2A9h	TV_H+9	2D9h	TV_H+57
22Ah	TV_L+10	25Ah	TV_L+58	2AAh	TV_H+10	2DAh	TV_H+58
22Bh	TV_L+11	25Bh	TV_L+59	2ABh	TV_H+11	2DBh	TV_H+59
22Ch	TV_L+12	25Ch	TV_L+60	2ACh	TV_H+12	2DCh	TV_H+60
22Dh	TV_L+13	25Dh	TV_L+61	2ADh	TV_H+13	2DDh	TV_H+61
22Eh	TV_L+14	25Eh	TV_L+62	2AEh	TV_H+14	2DEh	TV_H+62
22Fh	TV_L+15	25Fh	TV_L+63	2AFh	TV_H+15	2DFh	TV_H+63
230h	TV_L+16	260h	TV_L+64	2B0h	TV_H+16	2E0h	TV_H+64
231h	TV_L+17	261h	TV_L+65	2B1h	TV_H+17	2E1h	TV_H+65
232h	TV_L+18	262h	TV_L+66	2B2h	TV_H+18	2E2h	TV_H+66
233h	TV_L+19	263h	TV_L+67	2B3h	TV_H+19	2E3h	TV_H+67
234h	TV_L+20	264h	TV_L+68	2B4h	TV_H+20	2E4h	TV_H+68
235h	TV_L+21	265h	TV_L+69	2B5h	TV_H+21	2E5h	TV_H+69
236h	TV_L+22	266h	TV_L+70	2B6h	TV_H+22	2E6h	TV_H+70
237h	TV_L+23	267h	TV_L+71	2B7h	TV_H+23	2E7h	TV_H+71
238h	TV_L+24	268h	TV_L+72	2B8h	TV_H+24	2E8h	TV_H+72
239h	TV_L+25	269h	TV_L+73	2B9h	TV_H+25	2E9h	TV_H+73
23Ah	TV_L+26	26Ah	TV_L+74	2BAh	TV_H+26	2EAh	TV_H+74
23Bh	TV_L+27	26Bh	TV_L+75	2BBh	TV_H+27	2EBh	TV_H+75
23Ch	TV_L+28	26Ch	TV_L+76	2BCh	TV_H+28	2ECh	TV_H+76
23Dh	TV_L+29	26Dh	TV_L+77	2BDh	TV_H+29	2EDh	TV_H+77
23Eh	TV_L+30	26Eh	TV_L+78	2BEh	TV_H+30	2EEh	TV_H+78
23Fh	TV_L+31	26Fh	TV_L+79	2BFh	TV_H+31	2EFh	TV_H+79
240h	TV_L+32	**270h**	**I0**	2C0h	TV_H+32	**2F0h**	**I0**
241h	TV_L+33	**271h**	**I1**	2C1h	TV_H+33	**2F1h**	**I1**
242h	TV_L+34	**272h**	**I2**	2C2h	TV_H+34	**2F2h**	**I2**
243h	TV_L+35	**273h**	**I3**	2C3h	TV_H+35	**2F3h**	**I3**
244h	TV_L+36	**274h**	**Q0**	2C4h	TV_H+36	**2F4h**	**Q0**
245h	TV_L+37	**275h**	**Q1**	2C5h	TV_H+37	**2F5h**	**Q1**
246h	TV_L+38	**276h**	**Q2**	2C6h	TV_H+38	**2F6h**	**Q2**
247h	TV_L+39	**277h**	**Q3**	2C7h	TV_H+39	**2F7h**	**Q3**
248h	TV_L+40	**278h**	**Temp_1**	2C8h	TV_H+40	**2F8h**	**Temp_1**
249h	TV_L+41	**279h**	**Temp_2**	2C9h	TV_H+41	**2F9h**	**Temp_2**
24Ah	TV_L+42	**27Ah**	**SMB1**	2CAh	TV_H+42	**2FAh**	**SMB1**
24Bh	TV_L+43	**27Bh**	**SMB2**	2CBh	TV_H+43	**2FBh**	**SMB2**
24Ch	TV_L+44	**27Ch**		2CCh	TV_H+44	**2FCh**	
24Dh	TV_L+45	**27Dh**		2CDh	TV_H+45	**2FDh**	
24Eh	TV_L+46	**27Eh**		2CEh	TV_H+46	**2FEh**	
24Fh	TV_L+47	**27Fh**		2CFh	TV_H+47	**2FFh**	

Bank 4 Bank 5

FIGURE 2.3 Continued

In the other PLC scan cycles following the first one, it is reset. The special memory bit “SCNOSC” is arranged to work as a “scan oscillator”. This means that in one PLC scan cycle this special bit will hold the value of “0”, in the next one the value of “1”, in the next one the value of “0”, and so on. This will keep on going for every PLC scan cycle.

320h	CV_L	350h	CV_L+48
321h	CV_L+1	351h	CV_L+49
322h	CV_L+2	352h	CV_L+50
323h	CV_L+3	353h	CV_L+51
324h	CV_L+4	354h	CV_L+52
325h	CV_L+5	355h	CV_L+53
326h	CV_L+6	356h	CV_L+54
327h	CV_L+7	357h	CV_L+55
328h	CV_L+8	358h	CV_L+56
329h	CV_L+9	359h	CV_L+57
32Ah	CV_L+10	35Ah	CV_L+58
32Bh	CV_L+11	35Bh	CV_L+59
32Ch	CV_L+12	35Ch	CV_L+60
32Dh	CV_L+13	35Dh	CV_L+61
32Eh	CV_L+14	35Eh	CV_L+62
32Fh	CV_L+15	35Fh	CV_L+63
330h	CV_L+16	360h	CV_L+64
331h	CV_L+17	361h	CV_L+65
332h	CV_L+18	362h	CV_L+66
333h	CV_L+19	363h	CV_L+67
334h	CV_L+20	364h	CV_L+68
335h	CV_L+21	365h	CV_L+69
336h	CV_L+22	366h	CV_L+70
337h	CV_L+23	367h	CV_L+71
338h	CV_L+24	368h	CV_L+72
339h	CV_L+25	369h	CV_L+73
33Ah	CV_L+26	36Ah	CV_L+74
33Bh	CV_L+27	36Bh	CV_L+75
33Ch	CV_L+28	36Ch	CV_L+76
33Dh	CV_L+29	36Dh	CV_L+77
33Eh	CV_L+30	36Eh	CV_L+78
33Fh	CV_L+31	36Fh	CV_L+79
340h	CV_L+32	**370h**	**I0**
341h	CV_L+33	**371h**	**I1**
342h	CV_L+34	**372h**	**I2**
343h	CV_L+35	**373h**	**I3**
344h	CV_L+36	**374h**	**Q0**
345h	CV_L+37	**375h**	**Q1**
346h	CV_L+38	**376h**	**Q2**
347h	CV_L+39	**377h**	**Q3**
348h	CV_L+40	**378h**	**Temp_1**
349h	CV_L+41	**379h**	**Temp_2**
34Ah	CV_L+42	**37Ah**	**SMB1**
34Bh	CV_L+43	**37Bh**	**SMB2**
34Ch	CV_L+44	**37Ch**	
34Dh	CV_L+45	**37Dh**	
34Eh	CV_L+46	**37Eh**	
34Fh	CV_L+47	**37Fh**	

Bank 6

3A0h	CV_H	3D0h	CV_H+48
3A1h	CV_H+1	3D1h	CV_H+49
3A2h	CV_H+2	3D2h	CV_H+50
3A3h	CV_H+3	3D3h	CV_H+51
3A4h	CV_H+4	3D4h	CV_H+52
3A5h	CV_H+5	3D5h	CV_H+53
3A6h	CV_H+6	3D6h	CV_H+54
3A7h	CV_H+7	3D7h	CV_H+55
3A8h	CV_H+8	3D8h	CV_H+56
3A9h	CV_H+9	3D9h	CV_H+57
3AAh	CV_H+10	3DAh	CV_H+58
3ABh	CV_H+11	3DBh	CV_H+59
3ACh	CV_H+12	3DCh	CV_H+60
3ADh	CV_H+13	3DDh	CV_H+61
3AEh	CV_H+14	3DEh	CV_H+62
3AFh	CV_H+15	3DFh	CV_H+63
3B0h	CV_H+16	3E0h	CV_H+64
3B1h	CV_H+17	3E1h	CV_H+65
3B2h	CV_H+18	3E2h	CV_H+66
3B3h	CV_H+19	3E3h	CV_H+67
3B4h	CV_H+20	3E4h	CV_H+68
3B5h	CV_H+21	3E5h	CV_H+69
3B6h	CV_H+22	3E6h	CV_H+70
3B7h	CV_H+23	3E7h	CV_H+71
3B8h	CV_H+24	3E8h	CV_H+72
3B9h	CV_H+25	3E9h	CV_H+73
3BAh	CV_H+26	3EAh	CV_H+74
3BBh	CV_H+27	3EBh	CV_H+75
3BCh	CV_H+28	3ECh	CV_H+76
3BDh	CV_H+29	3EDh	CV_H+77
3BEh	CV_H+30	3EEh	CV_H+78
3BFh	CV_H+31	3EFh	CV_H+79
3C0h	CV_H+32	**3F0h**	**I0**
3C1h	CV_H+33	**3F1h**	**I1**
3C2h	CV_H+34	**3F2h**	**I2**
3C3h	CV_H+35	**3F3h**	**I3**
3C4h	CV_H+36	**3F4h**	**Q0**
3C5h	CV_H+37	**3F5h**	**Q1**
3C6h	CV_H+38	**3F6h**	**Q2**
3C7h	CV_H+39	**3F7h**	**Q3**
3C8h	CV_H+40	**3F8h**	**Temp_1**
3C9h	CV_H+41	**3F9h**	**Temp_2**
3CAh	CV_H+42	**3FAh**	**SMB1**
3CBh	CV_H+43	**3FBh**	**SMB2**
3CCh	CV_H+44	**3FCh**	
3CDh	CV_H+45	**3FDh**	
3CEh	CV_H+46	**3FEh**	
3CFh	CV_H+47	**3FFh**	

Bank 7

FIGURE 2.3 Continued

Let us now consider the four reference timing signals, namely T_2ms, T_10ms, T_100ms, and T_1s. As will be explained later, timer TMR6 of PIC16F1847 is set up to count ¼ of the 32-MHz oscillator signal, i.e., 8 MHz with a prescaler arranged to divide the signal to 64. Then the TMR6 interrupt flag, i.e., TMR6IF, will be set at every 1 ms. When TMR6IF is set, Boolean variables T_2ms, T_10ms, T_100ms, and T_1s will be processed within the "ISR" to obtain timing signals with periods

420h	M0	450h	M48
421h	M1	451h	M49
422h	M2	452h	M50
423h	M3	453h	M51
424h	M4	454h	M52
425h	M5	455h	M53
426h	M6	456h	M54
427h	M7	457h	M55
428h	M8	458h	M56
429h	M9	459h	M57
42Ah	M10	45Ah	M58
42Bh	M11	45Bh	M59
42Ch	M12	45Ch	M60
42Dh	M13	45Dh	M61
42Eh	M14	45Eh	M62
42Fh	M15	45Fh	M63
430h	M16	460h	M64
431h	M17	461h	M65
432h	M18	462h	M66
433h	M19	463h	M67
434h	M20	464h	M68
435h	M21	465h	M69
436h	M22	466h	M70
437h	M23	467h	M71
438h	M24	468h	M72
439h	M25	469h	M73
43Ah	M26	46Ah	M74
43Bh	M27	46Bh	M75
43Ch	M28	46Ch	M76
43Dh	M29	46Dh	M77
43Eh	M30	46Eh	M78
43Fh	M31	46Fh	M79
440h	M32	**470h**	**I0**
441h	M33	**471h**	**I1**
442h	M34	**472h**	**I2**
443h	M35	**473h**	**I3**
444h	M36	**474h**	**Q0**
445h	M37	**475h**	**Q1**
446h	M38	**476h**	**Q2**
447h	M39	**477h**	**Q3**
448h	M40	**478h**	**Temp_1**
449h	M41	**479h**	**Temp_2**
44Ah	M42	**47Ah**	**SMB1**
44Bh	M43	**47Bh**	**SMB2**
44Ch	M44	**47Ch**	
44Dh	M45	**47Dh**	
44Eh	M46	**47Eh**	
44Fh	M47	**47Fh**	

Bank 8

4A0h	M80	4D0h	T_Q0
4A1h	M81	4D1h	T_Q1
4A2h	M82	4D2h	T_Q2
4A3h	M83	4D3h	T_Q3
4A4h	M84	4D4h	T_Q4
4A5h	M85	4D5h	T_Q5
4A6h	M86	4D6h	T_Q6
4A7h	M87	4D7h	T_Q7
4A8h	M88	4D8h	T_Q8
4A9h	M89	4D9h	T_Q9
4AAh	M90	4DAh	C_Q0
4ABh	M91	4DBh	C_Q1
4ACh	M92	4DCh	C_Q2
4ADh	M93	4DDh	C_Q3
4AEh	M94	4DEh	C_Q4
4AFh	M95	4DFh	C_Q5
4B0h	M96	4E0h	C_Q6
4B1h	M97	4E1h	C_Q7
4B2h	M98	4E2h	C_Q8
4B3h	M99	4E3h	C_Q9
4B4h	M100	4E4h	C_QD0
4B5h	M101	4E5h	C_QD1
4B6h	M102	4E6h	C_QD2
4B7h	M103	4E7h	C_QD3
4B8h	M104	4E8h	C_QD4
4B9h	M105	4E9h	C_QD5
4BAh	M106	4EAh	C_QD6
4BBh	M107	4EBh	C_QD7
4BCh	M108	4ECh	C_QD8
4BDh	M109	4EDh	C_QD9
4BEh	M110	4EEh	
4BFh	M111	4EFh	
4C0h	M112	**4F0h**	**I0**
4C1h	M113	**4F1h**	**I1**
4C2h	M114	**4F2h**	**I2**
4C3h	M115	**4F3h**	**I3**
4C4h	M116	**4F4h**	**Q0**
4C5h	M117	**4F5h**	**Q1**
4C6h	M118	**4F6h**	**Q2**
4C7h	M119	**4F7h**	**Q3**
4C8h	M120	**4F8h**	**Temp_1**
4C9h	M121	**4F9h**	**Temp_2**
4CAh	M122	**4FAh**	**SMB1**
4CBh	M123	**4FBh**	**SMB2**
4CCh	M124	**4FCh**	
4CDh	M125	**4FDh**	
4CEh	M126	**4FEh**	
4CFh	M127	**4FFh**	

Bank 9

FIGURE 2.3 Continued

of 2 milliseconds, 10 milliseconds, 100 milliseconds, and 1 second, respectively. Timing diagrams of the reference timing signals T_2ms, T_10ms, T_100ms, and T_1s are depicted in Figure 2.8. Note that the evaluation of TMR6 is independent from PLC scan cycles. When the PLC is switched on, four reference timing signals (clock pulses), namely T_2ms, T_10ms, T_100ms, and T_1s, will start their operation automatically as shown in Figure 2.8.

520h	drum_TVL	550h	MB2	5A0h	LPF	5D0h	AI0L
521h	drum_TVL+1	551h	step_1.TL	5A1h	LPF+1	5D1h	AI0H
522h	drum_TVL+2	552h	step_1.TH	5A2h	LPF+2	5D2h	AI1L
523h	drum_TVL+3	553h	step_2.TL	5A3h	LPF+3	5D3h	AI1H
524h	drum_TVL+4	554h	step_2.TH	5A4h	LPF+4	5D4h	AI2L
525h	drum_TVL+5	555h	step_3.TL	5A5h	LPF+5	5D5h	AI2H
526h	drum_TVL+6	556h	step_3.TH	5A6h	LPF+6	5D6h	AI3L
527h	drum_TVL+7	557h	step_4.TL	5A7h	LPF+7	5D7h	AI3H
528h	drum_TVL+8	558h	step_4.TH	5A8h	LPF+8	5D8h	step_16.TH
529h	drum_TVL+9	559h	step_5.TL	5A9h	LPF+9	5D9h	step_17.TL
52Ah	drum_TVL+10	55Ah	step_5.TH	5AAh	LPF+10	5DAh	step_17.TH
52Bh	drum_TVL+11	55Bh	step_6.TL	5ABh	LPF+11	5DBh	step_18.TL
52Ch	drum_TVL+12	55Ch	step_6.TH	5ACh	LPF+12	5DCh	step_18.TH
52Dh	drum_TVL+13	55Dh	step_7.TL	5ADh	LPF+13	5DDh	step_19.TL
52Eh	drum_TVL+14	55Eh	step_7.TH	5AEh	LPF+14	5DEh	step_19.TH
52Fh	drum_TVL+15	55Fh	step_8.TL	5AFh	LPF+15	5DFh	step_20.TL
530h	drum_TVH	560h	step_8.TH	5B0h	LPF+16	5E0h	step_20.TH
531h	drum_TVH+1	561h	step_9.TL	5B1h	LPF+17	5E1h	step_21.TL
532h	drum_TVH+2	562h	step_9.TH	5B2h	LPF+18	5E2h	step_21.TH
533h	drum_TVH+3	563h	step_10.TL	5B3h	LPF+19	5E3h	step_22.TL
534h	drum_TVH+4	564h	step_10.TH	5B4h	LPF+20	5E4h	step_22.TH
535h	drum_TVH+5	565h	step_11.TL	5B5h	LPF+21	5E5h	step_23.TL
536h	drum_TVH+6	566h	step_11.TH	5B6h	LPF+22	5E6h	step_23.TH
537h	drum_TVH+7	567h	step_12.TL	5B7h	LPF+23	5E7h	step_24.TL
538h	drum_TVH+8	568h	step_12.TH	5B8h	LPF+24	5E8h	step_24.TH
539h	drum_TVH+9	569h	step_13.TL	5B9h	LPF+25	5E9h	HSC_B2
53Ah	drum_TVH+10	56Ah	step_13.TH	5BAh	LPF+26	5EAh	HSC_B3
53Bh	drum_TVH+11	56Bh	step_14.TL	5BBh	LPF+27	5EBh	
53Ch	drum_TVH+12	56Ch	step_14.TH	5BCh	LPF+28	5ECh	
53Dh	drum_TVH+13	56Dh	step_15.TL	5BDh	LPF+29	5EDh	
53Eh	drum_TVH+14	56Eh	step_15.TH	5BEh	LPF+30	5EEh	
53Fh	drum_TVH+15	56Fh	step_16.TL	5BFh	LPF+31	5EFh	
540h	drum_TQL	**570h**	**I0**	5C0h	LPF+32	**5F0h**	**I0**
541h	drum_TQH	**571h**	**I1**	5C1h	LPF+33	**5F1h**	**I1**
542h	drum_stepsL	**572h**	**I2**	5C2h	LPF+34	**5F2h**	**I2**
543h	drum_stepsH	**573h**	**I3**	5C3h	LPF+35	**5F3h**	**I3**
544h	drum_eventsL	**574h**	**Q0**	5C4h	LPF+36	**5F4h**	**Q0**
545h	drum_eventsH	**575h**	**Q1**	5C5h	LPF+37	**5F5h**	**Q1**
546h	drum_QL	**576h**	**Q2**	5C6h	LPF+38	**5F6h**	**Q2**
547h	drum_QH	**577h**	**Q3**	5C7h	LPF+39	**5F7h**	**Q3**
548h	drum_tmp	**578h**	**Temp_1**	5C8h	nAI0L	**5F8h**	**Temp_1**
549h	drum_tmpL	**579h**	**Temp_2**	5C9h	nAI0H	**5F9h**	**Temp_2**
54Ah	drum_tmpH	**57Ah**	**SMB1**	5CAh	nAI1L	**5FAh**	**SMB1**
54Bh	SF0	**57Bh**	**SMB2**	5CBh	nAI1H	**5FBh**	**SMB2**
54Ch	SF1	**57Ch**		5CCh	nAI2L	**5FCh**	
54Dh	SF2	**57Dh**		5CDh	nAI2H	**5FDh**	
54Eh	MB0	**57Eh**		5CEh	nAI3L	**5FEh**	
54Fh	MB1	**57Fh**		5CFh	nAI3H	**5FFh**	

Bank 10 Bank 11

FIGURE 2.3 Continued

Time delays are obtained by using one of these four reference timing signals. For example, if, say, we need 5 seconds' time delay, we can obtain it by counting the T_10ms signal 500 times (10 ms × 500 = 5,000 ms = 5 s) or by counting the T_100ms signal 50 times (100 ms × 50 = 5000 ms = 5 s). The counting process is carried out by using the rising edge signals instead of using the original reference timing signals. The time interval from one rising edge of a reference timing signal to the

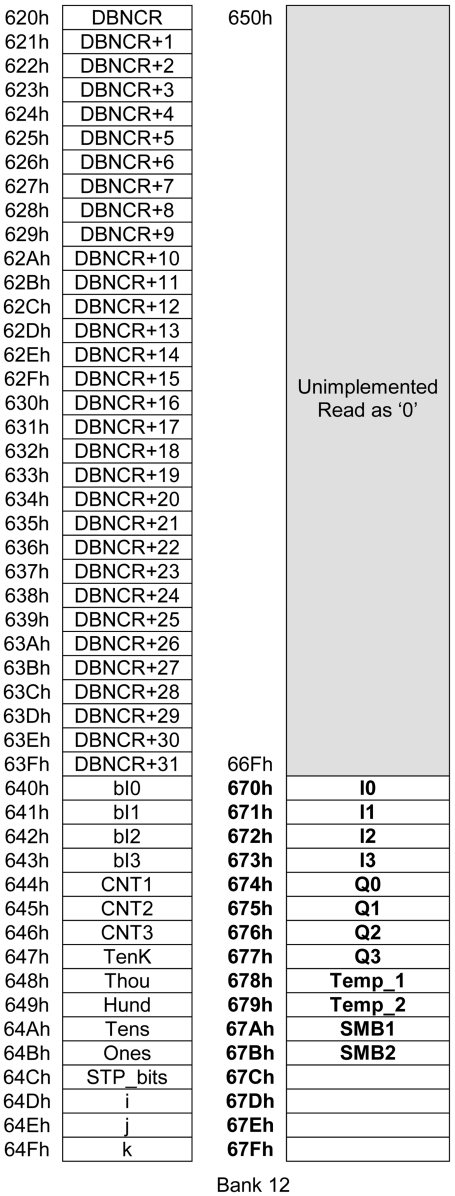


FIGURE 2.3 Continued

next one is equal to the period of that signal. As a result, in this project, rising edge signals re_T2ms, re_T10ms, re_T100ms, and re_T_1s are obtained from reference timing signals T_2ms, T_10ms, T_100ms, and T_1s, respectively, to be used in timing-related functions. Figure 2.9 shows timing diagrams of a reference timing signal (RTS) (T[period] = 2 ms, 10 ms, 100 ms, 1 s) and the rising edge signal of the RTS.