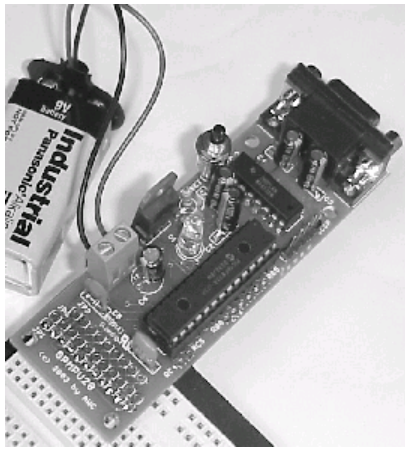
AWC

APP-II PIC
Development Kit



AWC
1279 FM 518 #2
Kemah, TX 77565
(281) 334-4341
http://www.awce.com
V1.3 7 Sep 2004

Table of Contents

Overview

The APP-II allows you to develop PIC code for the popular PIC16F87x microcontroller. The basic kit includes a special PIC16F873 (28 pin microprocessor) that operates at 20MHz and a 20MHz crystal. The full kit also includes an RS-I serial interface kit (pictured on the front cover). You'll find the RS-I manual included with that kit. If you did not get the full kit, you'll need some way to connect an RS-232 port from your computer to the APP-II chip. This requires some sort of level conversion (such as a MAX232 chip, which is what the RS-I uses). You can not directly connect a PC's RS-232 port to the APP-II.

If You Need Help

If you require assistance, please feel free to contact us. The best way to get support is via e-mail (stamp@al-williams.com). However, you may also call between 9AM - 4PM Central Time at (281) 334-4341. You can also fax to (281) 754-4462. Be sure to check out our Web page for updates at www.awce.com.

What Else You'll Need

In addition to the APP-II kit, you'll also need a few other easy to obtain items:

- A solderless breadboard or other substrate to build the circuit on.
- A 5V regulated power supply.
- MPLAB, the free software from Microchip (or another development tool that produces 8-bit hex files)
- A serial cable (a DB9 male to female, if you are using a PC).

Features

The PIC16F873 core of the APP-II has 5 channels of 10-bit A/D, up to 19 digital I/O, and a hardware serial port usable by your programs. The device has 192 bytes of RAM, 256 bytes of EEPROM, and 3836 program words.

Assembly

There are two versions of the APP-II. The experimenter's kit includes an RS-I kit and requires you to build your PIC circuit on a solderless breadboard or other medium. The standard version includes a PC board that holds the PIC, an optional power supply, and other required circuitry.

GPMPU28-Based Kit

If you have the GPMPU28 board – that is, you have the standard version – please refer to the GPMPU28 manual (included) for details about how to build the circuit. Please note, that the schematic differs slightly from the one in this manual (R3 is not present on the GPMPU28 board and is not needed). The GPMPU28 manual has the correct schematic.

Note that the board is slightly modified for the APP-II kit. Please follow these directions to modify the board for this kit:

- At the bottom of the board, there are three headers marked JP1, JP2, and JP3. Pin 1 is to the left (see the back of the GPMPU28 manual). The board has been modified so that pin 2 of JP1 and JP2, along with pin 1 of JP3 are isolated from their normal connections. Solder a two-pin header (included with the kit) into JP2 pin 1 and 2. This is the APP-II program jumper.

- On the right side of the board, there are two holes marked CTS and RB3 (the RB3 marking is under the CPU). You will install a 1K resistor in these holes. **IMPORTANT:** the resistor installs vertically. Place the body of the resistor flush with the RB3 hole in the board. Bend the opposite lead over and place it in the CTS hole (not the RB2 hole

which is adjacent). Solder. **Do not place the resistor body flush with the CTS hole – you will need the wire in CTS for the next step.**

- With ordinary hook up wire, form a connection from JP3 pin 1 to the resistor you soldered in the last step. Loop the wire around the wire lead in the CTS hole and solder. Be careful not to desolder the 1K resistor in the process.

Experimenter's Kit

If you kit includes an RS-I, it is in kit form, and you'll need to assemble it first. Refer to the RS-I manual included.

Next, build the circuit shown in Figure 1 on a solderless breadboard. Keep the wires to the resonator neat and short. The outer pins of the resonator are interchangeable. Be sure to connect power to the RS-I. You won't use the last two pins of the RS-I board in this project (although you can connect them to any of the APP-II's I/O pins for your own projects).

If you have trouble getting JP1 or the switch to stay in your breadboard, try spreading the pins that insert into the breadboard a bit with a pair of pliers. Also, you can use pliers to move the plastic insulator so that there is the same length of pins above and below the insulator, which can also help.

Testing

The APP-II is shipped with a test program already on board. Place the shorting cap (jumper) on JP1 (JP2 in the PCB version) and connect a voltmeter or a logic probe on RB0. When you apply 5V to the circuit, RB0 should go high for approximately one second and then go low for one second. The LED on the GPMPU board will also blink. This should repeat indefinitely. If this doesn't happen, disconnect the power immediately and check all connections.

Once the RB0 test is successful, connect your PC to the RS-I's connector. Start a terminal program (such as Hyperterminal, which comes with Windows). Make sure that the port you want to use is not in use, and that the complimentary port is also not in use (most PCs can't use COM1 and COM3 or COM2 and COM4 at the same time).

The rest of this discussion will assume you are using Hyperterminal. When you start Hyperterminal it will ask you to create a new connection. Call it APPII (or any name you like). The prompt will ask you to enter a phone number and will show a box that contains the ID of your modem. You can also select a direct COM port in this box, and that's what you'll do. Pick the COM Port you are using to connect to the APP-II.

Next, Hyperterminal will prompt you for serial port parameters. The APP-II uses 19200 baud (19.2 kBaud), 8 bits, 1 stop bit and no parity. Most importantly, you must set the handshaking to Hardware (some programs call this RTS/CTS handshaking).

Once you've set up the connection, you should be able to see hex numbers displaying on the terminal. This shows that your RS-232 connection is working. If you don't see the output (but RB0 is still pulsating) check the RS-I wiring and the settings on the terminal program (for example, some programs require you to use a connect command to start communications).

If you are using Hyperterminal, you should also bring up the File | Properties menu. From there, select the Settings tab and then press ASCII Setup. Make sure Echo typed characters locally is not checked (it shouldn't be). Also check Append line feeds to incoming line ends. Select TTY as the emulation type. Then dismiss the dialog boxes by pressing OK twice.

Congratulations! Your APP-II is working. Now you can do some programming.

Programming

To assist you further in programming, you can download a few sample HEX files from our Web site (www.awce.com). One of these is the test program that is running on the APP-II right now. To test the programming feature of the APP-II, remove the shorting cap (jumper) from JP1 (JP2 on the PCB version) and press the RESET button. On your terminal screen you should see a > character – this tells you the APP-II is ready to receive a hex file.

Suppose you select the 1st.hex from the Web site. If you look at this file with notepad or a similar text editor you will see that it is in hex format (the source code is in 1st.asm). You can send this hex file directly to the APP-II. For Hyperterminal, the easiest way to do this is to select the Transfer | Send Text File menu command. Just pick the hex file you want to download and press OK. You can also copy the hex code to the clipboard and then paste it to the terminal program.

The APP-II will echo the characters you send back to you so you can check its progress. If it encounters an error, the programming will stop with an @ character displayed on the terminal. A ! character appears when programming is complete and successful.

To run the new program, replace the jumper cap on JP1 (JP2 on the PCB version) and press the reset button. This is the basic steps you'll use to program the APP-II each time you want to run a program.

You can write your own programs using any language that generates 8 bit hex files like MPLAB from Microchip.

Internals

The APP-II is actually a PIC16F873 that operates at 20MHz. You can find documentation for this device at the Microchip Web site. However, the APP-II environment makes programming a little different. In particular:

- The top program address you can use is 0x0EFA.

- Your program must execute a GOTO within the first four instructions (starting at 0). In addition, no GOTOs should target locations 0 through 3. Usually, this space will contain a jump around an interrupt routine, anyway, so this is not a big restriction.

- Port B pin 3 is not available for your use.

- Although you can use the serial port in your own programs, you must be able to configure the two serial port pins for serial I/O to program. Therefore, we don't recommend you use the serial port for anything other than a serial port.

- For programming, the APP-I must use a 20MHz clock.

In addition, you can make calls to the APP-II's library routines to perform useful tasks. You can download APPII.INC from our Web site which has definitions for these routines along with examples of their use.

The calls you can make are:

- SERIALINIT – Initialize the serial port to 19.2Kbaud (assumes 20MHz clock). Changes register bank to 1.

- SERIALSEND – Sends the character in W to the serial port; returns with bank 0 selected.

- SERIALRCV – Receives a character from the serial port to W. Returns with bank 0 selected.

- HEXOUT – Writes the data in INDF (pointed to by FSR) to the terminal as a 2 digit hex number. Returns with bank 0 selected.

- HEXOUT0 – Writes the least significant 4 bits of W out as a single hex digit. Returns with bank 0 selected.

- DELAY – Delays the number of clocks specified in the W register (from 11 to 255). Does not count the cycle required to load W, so the actual delay is from 12 to 256 cycles (2.4uS to 51.2uS). Does not change bank.

- GETHEX – Reads two hex digits from the terminal and leaves the result in register 0x20. Returns with bank 0 selected.

Resources

http://www.awce.com/app2.htm – Examples and include files.

http://www.awce.com/classroom – Tutorials.

http://www.awce.com/seabass.htm - Basic Compiler for APP-II.

http://tutor.al-williams.com/picc1.htm - C compiler for APP-II.

http://www.microchip.com – Download manuals for the PIC and MPLAB for free.

http://www.piclist.com – Information and help about the PIC.

http://www.hilgraeve.com/htpe/ - Download the latest Hyperterminal; free for personal use.

http://hp.vector.co.jp/authors/VA002416/teraterm.html – A freeware terminal.

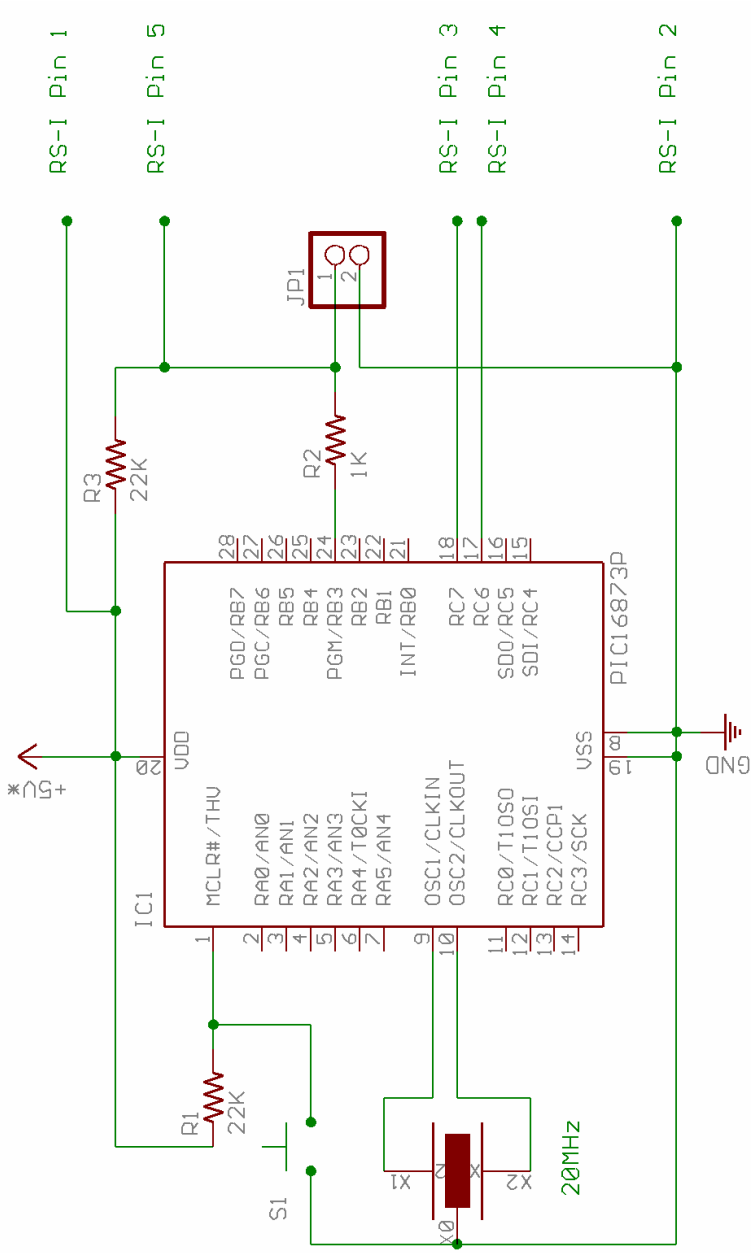


Figure 1. Experimenter's Version Schematic

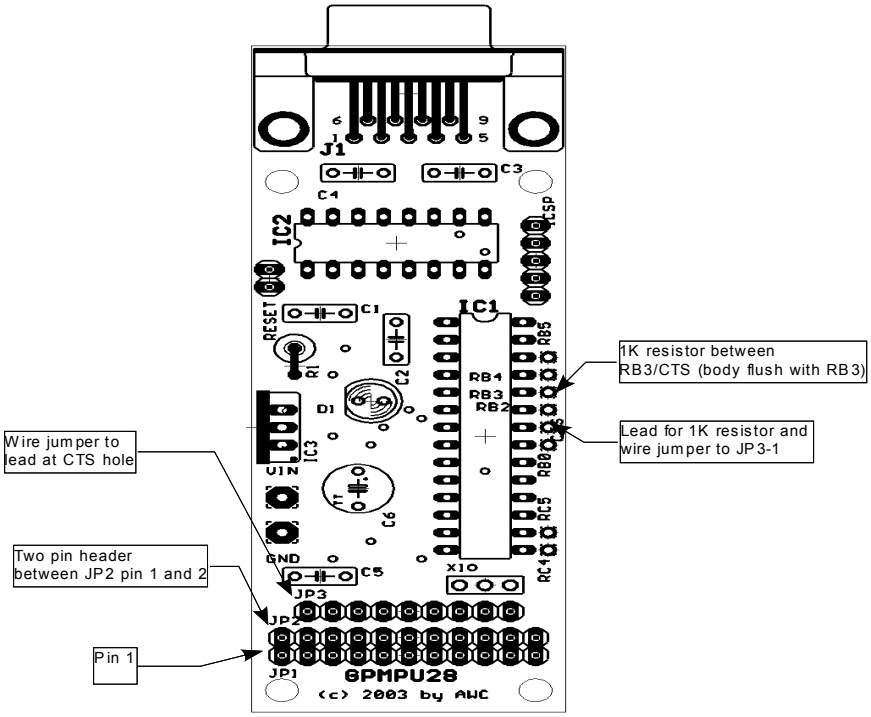


Figure 2. GPMPU28 Modifications

APP-II Include File

```
; Defines for APP-II routines
; Description: Set up RS232 port (19.2K baud)
; Call with bank= 0
; Returns with bank= 0
SerialSetup EQU 0x0F03

; Description: Send data in W to RS232
; Call with bank= Any
; Returns with bank= 0
SerialTransmit EQU 0x0F14

; Description: Read character from RS232 to W (waits)
; Echos character back to sender
; Call with bank= Any
; Returns with bank= 0
SerialReceive EQU 0x0F0E

; Description: Write byte at FSR to RS232 as ASCII hex
; Call with bank= Any
; Returns with bank= 0
Hexout EQU 0xF1A

; Description: Writes least significant 4 bits of W out as
ASCII hex
; Call with bank= Any
; Returns with bank=0
Hexout0 EQU 0xF1D

; Description: Delays # of cycles in W (11 to 255)
; does not count time to load W or PCLATH, if necessary
; Call with bank= Any
; Returns with bank= No change
Delay EQU 0xF23

; Description: Writes data at EEDATH:EEDATA to EEADRH:EEADR
; (Flash memory)
; Call with bank= Any
; Returns with bank= No change
FlashWrite EQU 0xFBF
```

```
; Description: Read word from EEADRH:EEADR to EEDATH:EEDATA
; (Flash memory)
; Call with bank=Any
; Returns with bank=2
FlashRead EQU 0xFCA

; Description: Read a hex byte from RS232 to W & location
0x20
; Call with bank=Any
; Returns with bank=0
; warning routine uses
; RAM 0x20
GetHexByte EQU 0xFAD
```

Notes