ON-CHIP DEBUGGER
SPECIFICATION

Information contained in this publication regarding device applications and the like is intended by way of suggestion only. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information. Use of Microchip’s products as critical components in life support systems is not authorized except with express written approval by Microchip.

© 2001 Microchip Technology Incorporated. All rights reserved.
The Microchip logo, name, PIC, PICmicro, PICMASTER, PICSTART, and PRO MATE are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries. MPLAB and Smart Serial are trademarks of Microchip Technology in the U.S.A. and other countries.
All product/company trademarks mentioned herein are the property of their respective companies.
ON-CHIP DEBUGGER
SPECIFICATION

Table of Contents

Chapter 1. Overview
  1.1 Introduction .......................................................................................... 5
  1.2 Highlights ............................................................................................. 5
  1.3 On-Chip Debugger Definition .............................................................. 5
  1.4 Debugger vs. Emulator ........................................................................ 6

Chapter 2. PIC16F87X Debugger Control
  2.1 Introduction .......................................................................................... 7
  2.2 Highlights ............................................................................................. 7
  2.3 Enabling/Disabling the On-Chip Debugger ......................................... 7
  2.4 Using the On-Chip Debugger Registers .............................................. 8
  2.5 Communicating with the Target Device ............................................. 9
  2.6 Resetting and Running Programs ...................................................... 9
  2.7 Halting Execution ................................................................................ 9
  2.8 Additional Considerations .................................................................. 11

Index .........................................................................................................13
On-Chip Debugger Specification
Chapter 1. Overview

1.1 Introduction

This chapter provides an overview of the on-chip debugger.

1.2 Highlights

Topics covered in this chapter are:
- On-Chip Debugger Definition
- Debugger vs. Emulator

1.3 On-Chip Debugger Definition

An on-chip debugger is special hardware and software that works with specific PICmicro® devices to give developers a low cost method for debugging their code. These PICmicro MCU’s must (1) contain special on-chip logic supporting debugging functionality and (2) provide In-Circuit Serial Programming™ (ICSP™) capabilities.

The ICSP capability allows in-circuit programming of the PICmicro MCU via specific hardware pins designated for this purpose. ICSP is described in detail in the In-Circuit Serial Programming (ICSP) Guide (DS30277). Additionally, ICSP-related technical briefs and application notes, as well as device programming specifications, may be found at our website (www.microchip.com).

The on-chip debug capability provides breakpointing, single stepping, and external breaking. Breaks are treated as a special form of interrupt that vector to debug executive software implemented by the developer. The same hardware pins used by the ICSP capability are preserved by the on-chip debug capability while servicing the debug break, allowing them to be used to implement communications between the debug executive software and a host processor.
1.4 Debugger vs. Emulator

Traditionally, embedded systems engineers use in-circuit emulators (ICE) to develop and debug their designs. The on-chip debug facilities in some PICmicro devices provide a low cost alternative to a more expensive ICE.

When implemented, the on-chip debugger logic is part of the actual microcontroller silicon. As a result, there is a trade-off between the cost of producing this silicon on every chip (even though only a few will actually use this logic) and the power of the logic itself. It would be possible to put an entire emulator with complex breakpoints and tracing capabilities on a chip, but the cost would be prohibitively high. Usually the on-chip debugger provides the means to set simple breakpoints, query the internal state of the chip, and single step through code.

A microcontroller utilizing its on-chip debug facilities has these benefits:

- Low cost
- Requires a minimum of extra hardware
- Does not require expensive sockets or adapters
- Able to debug a production line board

In order to have the above benefits, on-chip debuggers have these trade-offs compared to ICEs:

- Use of some target system resources such as pins, program and/or data memory, and stack space may mean that some portions of an embedded application cannot be debugged.
- Triggering and breakpointing are limited to the capabilities of the on-chip debug silicon.
- The target chip must actually be running with a clock and a supply voltage. Often an emulator probe can run without any external hardware.

Overall, an in-circuit emulator usually provides the most powerful debugging solution, but as chips get faster and more complex, emulators require increasingly expensive high speed logic and high priced adapters to connect to the denser chip packages.
Chapter 2. PIC16F87X Debugger Control

2.1 Introduction

This chapter describes the hardware and software requirements for on-chip debugging using the PIC16F87X.

2.2 Highlights

Topics covered in this chapter are:

- Enabling/Disabling the On-Chip Debugger
- Using the On-Chip Debugger Registers
- Communicating with the Target Device
- Resetting and Running Programs
- Halting Execution
- Additional Considerations

2.3 Enabling/Disabling the On-Chip Debugger

The configuration bit BKBUG (bit 11 in the configuration word) is used to enable/disable the on-chip debugger (Figure 2.1).

See the programming specifications for PIC16C87X devices (DS39025) for more information on programming the configuration word.

Figure 2.1: CONFIGURATION WORD

<table>
<thead>
<tr>
<th>CP1</th>
<th>CP0</th>
<th>BKBUG</th>
<th>WRT</th>
<th>CPD</th>
<th>LVP</th>
<th>BODEN</th>
<th>CP1</th>
<th>CP0</th>
<th>PWRT</th>
<th>WDTE</th>
<th>F0SC1</th>
<th>F0SC0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit13</td>
<td>bit0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

bit 11: BKBUG: On-chip Debugger Mode (This bit documented as reserved in data sheet)

1 = On-chip debugger functions not enabled
0 = On-chip debugger functional.
2.4 Using the On-Chip Debugger Registers

There are two registers associated with debugger control: the ICKBUG and BIGBUG registers (Figure 2.2 and Figure 2.3.) Between these two registers, there are 3 bits for debugger control and 13 bits for breakpoint address specification.

**INBUG** - Set/cleared automatically when entering/exiting debug executive. A four (4) instruction cycle delay will prevent the new state of zero from being recognized by the core to allow a safe debugger exit. When set, interrupts are disabled (regardless of the state of the GIE bit) and the debugger HALT is disabled. This prevents reentry into the debugger while communicating with computer software.

**FREEZ** - When FREEZ is set, TMR0, TMR1 and TMR2 and their prescalers freeze at a break (suspend counting). The SSP, A/D converter and USART state machines will also suspend upon the assertion of the FREEZ bit. Status flag bits that are altered by register read operations are unaffected while FREEZ is asserted.

**SSTEP** - Enables single step operation. After a return to normal operation (INBUG=0), if SSTEP is set, the on-chip logic will allow one user instruction to execute before the on-chip debugger is reentered.

**BKAnn** - Break address nn. These bits are cleared at POR. No other reset clears the bits, (i.e., a MCLR cannot change the breakpoint address value).

---

**Figure 2.2: ICKBUG Register (Address 18Eh)**

<table>
<thead>
<tr>
<th>bit 7:</th>
<th>INBUG: On-chip debugger activity status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 = Device is executing on-chip debugger code</td>
</tr>
<tr>
<td></td>
<td>0 = Device is executing user code</td>
</tr>
<tr>
<td>(This bit is read only; set from on-chip halt or breakpoint occurrence; only clear by RETURN)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 6:</th>
<th>FREEZ: on-chip debugger freeze mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 = Peripherals will freeze when INBUG=1</td>
</tr>
<tr>
<td></td>
<td>0 = Peripherals will not freeze when INBUG=1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 5:</th>
<th>SSTEP: Single Step Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 = Program will execute 1 instruction word of user code upon RETURN from debug code</td>
</tr>
<tr>
<td></td>
<td>0 = Program will execute multiple instruction words of user code</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 4-0:</th>
<th>BKA: Breakpoint Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When writing, represents the requested breakpoint address.</td>
</tr>
<tr>
<td></td>
<td>When reading, represents the value of the PC at last breakpoint, halt, or single step operation.</td>
</tr>
</tbody>
</table>
2.5 Communicating with the Target Device

The debug executive establishes the communication channel between the target device (RB<7:6>) and computer software (e.g., MPLAB IDE.) Since the debugger will issue read-modify-write instructions (BSF, BCF) on PORTB for communication with the debugger interface module, addresses 0x106 and 0x186 should be used to modify PORTB<7:6> and TRISB<7:6> so that bits <5:0> are not corrupted. If the user requests a read or a write of PORTB or TRISB, addresses 0x006 and 0x086 can be used to access the RB<5:0> portions of the register.

2.6 Resetting and Running Programs

In the debug environment, the target device reset pin can be activated by the module/target connector or by the target system resources. After reset, the target device will begin running the target program. To begin debugger execution, initiate a hardware halt (on I/O pin.)

Note: A hardware halt after reset will execute location 0x0000 and then skip to 0x0001. Therefore, a NOP is recommended at location 0x0000.

2.7 Halting Execution

If BKBUG is programmed to 0, the INBUG flag is clear, and a halt signal occurs, normal program execution is halted and the on-chip debugger is entered. A halt signal will be generated when either:

- RB6 transitions from high to low
- the current PC value matches the value of BKA<12:0>
- the STEP bit is set
When normal program execution is halted, the current instruction will be executed and then the program will be stopped at the address of the next instruction. The processor will go through what appears to be a normal interrupt cycle, except it will not disturb the contents of the GIE bit. On-chip debugger logic will control sequencing of the key interrupt signals in the core logic.

Once in debug mode, the INBUG flag is set. The PC is pushed onto the stack and stored in BKAnn. This allows proper reporting of the current state of the PC (next instruction to be executed) when the on-chip debugger is entered. The PC is then set to 0x2004.

As the processor vectors to 0x2004, the CPU will fetch and execute the instruction at this address. Therefore, address 2004 must be programmed with a GOTO opcode, where opcode is pointing to the beginning address of the debug code. Then the on-chip debugger routine will commence. PCLATH will be undisturbed, and the PC at breakpoint is at the top of the stack.

The debugger software will use a RETURN instruction to return to the mainline code. The RETURN will cause the INBUG bit to be cleared and will also release peripheral freeze at the proper time, if the FREEZ bit had been set.

### 2.7.1 Halting Execution by I/O Pin

For the conditions specified in Section 2.7, an edge detection circuit will generate a HALT signal pulse when RB6 transitions from high to low. This is also known as external or hardware halting.

![Figure 2.4: On-chip Debugger Execution Sequence](image-url)
2.7.2 Halting Execution by Address Breakpoint

For the conditions specified in Section 2.7, another HALT method is by address breakpoint. The ICKBUG and BIGBUG registers contain a 13-bit value (BKAnn) that is compared against the current PC value. When the values are equal, then the circuit will generate a halt signal on that cycle.

Power-up initializes the state of the ICKBUG and BIGBUG registers to 0x0000, equal to the reset vector.

Note: A hardware halt after reset will execute location 0x0000 and then skid to 0x0001. Therefore, a NOP is recommended at location 0x0000.

Disabling the breakpoints is implemented by setting a breakpoint address that lies in the address space of the on-chip debugger (ex: 0x1F00.)

2.7.3 Halting Execution by Single Stepping

For the conditions specified in Section 2.7, another HALT method is by single step. Upon exit of the debug routine by the return instruction, if the SSTEP bit is set, the on-chip debugger logic will generate a HALT signal timed to allow one instruction execution. After this, the device will reenter the on-chip debugger routine in the same fashion as the external (I/O pin) halt.

2.8 Additional Considerations

In addition to the logic required to support the on-chip debugger as described in the above sections, there are several issues of which to be aware.

- Watchdog Timer will still be functional with no special modes associated with on-chip debugger.
- The reset, POR and BOR logic will still be functional with no special modes associated with the on-chip debugger.
Index

A
A/D ................................................. 8
Address 18E ................................. 8
Address 18F .................................. 9
Address 2004 .................................. 10
Address 2007 ................................. 7

B
BIGBUG Register ......................... 9, 11
BKAnn ........................................ 8, 9, 10
BKBUG ......................................... 7, 9
BOR ........................................... 11

C
Communicating with the Target Device ... 9
Configuration Word ...................... 7

D
Debugger vs. Emulator .................... 6

E
Enabling/Disabling On-Chip Debugger ... 7
Execution Sequence ..................... 10

F
FREEZ ......................................... 8, 10

G
GIE Bit ......................................... 8, 10

H
Halting Execution
  Breakpoint .................................. 11
  I/O Pin .................................... 9, 10
  Single Stepping .......................... 11
Hardware Considerations .............. 11

I
I/O Pin ........................................ 9
ICE ......................................... 6
ICKBUG Register ....................... 8, 11
ICSP ........................................ 5
INBUG .................................. 8, 9, 10

M
MCLR ........................................ 8

O
On-Chip Debugger Definition .......... 5

On-Chip Debugger Registers .......... 8
On-Chip Debugger, Enabling/Disabling ... 7
Overview .................................. 5

P
PIC16F87X Debugger Control ........... 7
POR ........................................... 8, 11
PORTB ........................................ 9

R
RB6 ........................................... 9, 10
Register
  BIGBUG .................................. 9, 11
  ICKBUG .................................. 8, 11
Reset ........................................ 11
Resetting Programs ..................... 9
Running Programs ....................... 9

S
SSP ........................................... 8
SSTEP ....................................... 8, 9, 11

T
Timers ...................................... 8

U
USART ....................................... 8
Using On-Chip Debugger Registers .. 8

W
Watchdog Timer ......................... 11