Case Number: **01431152**

Subject: **PIC18F47Q43 silicon bug PUSHL opcode**

Case Number 01431152	IDE Name
Original Case Number	IDE Version
Case Reason Hardware/Firmware Support	Compiler
Customer Subject PIC18F47Q43 silicon bug PUSHL opcode	Compiler Version
Date/Time Opened 3/30/2024 2:00 PM	Compiler Edition
Status Resolution Proposed	HPA Activation Key
Urgency Critical	Stack / Library Name / App Note
Primary Target Device PIC18F47Q43	Programmer / Debugger
Additional Microchip Devices	Case Category Microcontrollers and Processors
Non-Microchip Devices Involved	Case Sub-Category 8-bit Microcontrollers
Peripheral/Component	Disti Client Account Name
Check All Devices?	IsWorkingDirectlyCloudPartner
	OtherCloudServices

Description

The extended instruction set opcode PUSHL fails using some literal value:

PUSHL 0xEF
PUSHL 0xEE
PUSHL 0xED
PUSHL 0xEC
PUSHL 0xEB
PUSHL 0xE7
PUSHL 0xE6
PUSHL 0xE5
PUSHL 0xE4
PUSHL 0xE3
PUSHL 0xDE
PUSHL 0xDC
PUSHL 0xDB

All of these corrupt the FSR2 register, some fail to push the literal value.

Exactly how the opcode fails depends on what value is in the WREG and what bank is selected.

Our unit test can be found here:

https://github.com/dsoze1138/MPLABX_pic-as_examples/tree/master/18F47Q43_UART_picas_v615.X

Design Stage Design
Application Details This is a execution time dependent subsystem in a image processing test system. Selection of the controller vendor depends on the resolution of this issue.

Phone Access Number 0030054192
Customer Can Call in From: 3/30/2024 6:00 PM
Note All times shown are based on your Time Zone settings. View our Support Phone Numbers Here
Contact Name Dan Soze
Case Origin Community Portal

Proposed Resolution

Hi Dan,

Thank you for reaching out to Microchip!

By taking a look in the SFR, at these addresses that you have mentioned in your comment are the indirect operands PREINC, POSTINC, POSTDEC, PLUSW. Here is an explanation of what each of them does:

10.4.3.2 FSR Registers and POSTINC, POSTDEC, PREINC and PLUSW

In addition to the INDF operand, each FSR register pair also has four additional indirect operands. Like INDF, these are "virtual" registers that cannot be directly read or written. Accessing these registers actually accesses the location to which the associated FSR register pair points, and also performs a specific action on the FSR value. They are:

- POSTDEC: accesses the location to which the FSR points, then automatically decrements the FSR by 1 afterwards
- POSTINC: accesses the location to which the FSR points, then automatically increments the FSR by 1 afterwards
- PREINC: automatically increments the FSR by one, then uses the location to which the FSR points in the operation
- PLUSW: adds the signed value of the W register (range of -127 to 128) to that of the FSR and uses the location to which the result points in the operation.

Using the PUSHL instruction with one of these operands will also execute the respective operation, so that is why you may see unexpected behavior.

For instance, when you write PUSHL 0xDC, it will execute the PREINC function, which increments the FSR by one and then writes the respective literal value to the address pointed by FSR.

Thank you,

Kind regards,

Ana

Created by: Dan Soze (4/1/2024, 10:15)

Hi Ana,
The proposed resolution does not address the issue I have raised.

The PUSHL opcode is documented to write a LITERAL to memory addressed by FSR2 then decrement FSR2.

What your response describes is what the implementation in the PIC18F47Q43 silicon actually does. This is not what is described in the data sheet.

The issue that Microchip must answer is why the data sheet or errata for the PIC18F47Q43 fails to contain any information about why the PUSHL opcode does not act as documented in the data sheet.

Created by : Ana Nitu (4/2/2024, 05:46)

Hi Dan,

The proposed resolution was indeed the description of what the silicon actually does. I have informed the internal development team about this behavior, and they will address it.

We can keep this case open until the internal team comes back with an answer regarding how the situation will be addressed if you would like.

Could you let me know if this is a blocking issue for your project?

Best regards,

Ana

Created by : Dan Soze (4/2/2024, 11:22)

Hello Ana,

Thank you for being so responsive to our request.

As this issue must be handled outside of your range of responsibilities we are putting the selection of Microchip controllers on hold pending action on this issue.

Thanks again for your help.

Created by : Ana Nitu (5/16/2024, 00:35)

Hi Dan,

I've discussed with the business unit and we will address the situation.

Is there anything else that I can help you with or can we close this case?

Thank you,

Ana

Created by : Dan Soze (5/18/2024, 22:59)

As of this date Microchip has not yet described what is any action they will pursue in addressing this issue.

I would prefer that this case remain open until someone with the authority has actually done something, even if that is to announce that no action of any kind will be taken.

Created by : Ana N. (8/5/2024, 23:08)

Hi Dan,

Thank you for your patience!

After discussing with the business unit, we've come up with the conclusion that the behavior that you are encountering is mentioned in the datasheet (DS40002147H – page 76) at *Chapter 9.5.3.3 Operations by FSRs on FSRs*:

9.5.3.3 Operatons by FSRs on FSRs

> Indirect Addressing operations that target other FSRs or virtual registers represent special cases. For example, using an FSR to point to one of the virtual registers will not result in successful operations. As a specifc case, assume that FSR0H:FSR0L contains the address of INDF1. Attempts to read the value of the INDF1 using INDF0 as an operand will return 00h. Attempts to write to INDF1 using INDF0 as the operand will result in a NOP.
>
> On the other hand, using the virtual registers to write to an FSR pair may not occur as planned. In these cases, the value will be written to the FSR pair but without any incrementing or decrementing. Thus, writing to either the INDF2 or POSTDEC2 register will write the same value to FSR2H:FSR2L.
>
> Since the FSRs are physical registers mapped in the SFR space, they can be manipulated through all direct operations. Users need to proceed cautiously when working on these registers, particularly if their code uses Indirect Addressing.
>
> Similarly, operations by Indirect Addressing are permitted on all other SFRs. Users need to exercise the appropriate caution that they do not inadvertently change settings that might affect the operation of the device.

Let me know if there is anything else that I can help you with!

Kind regards,

Ana

Created by : Ana N. (8/8/2024, 03:06) Proposed Resolution:

Hi Dan,

Thank you for reaching out to Microchip!

Considering that your initial requests and questions were solved, I will move this case to the "Resolution Proposed" status. Even though it is marked close, it won't close for another two weeks, so please feel free to reply if you would like to add more information.

For other issues, please open another case on our Microchip Support page.

Thank you,

Kind regards,

Ana

Created by : Dan Soze (8/9/2024, 01:12) The proposed resolution does not described the observed behavior in any way that is comprehensible for anyone of modest experience. In short it is useless to for the type of developer using this family of Microchip controllers for the first time.

Created by : Dan Soze (8/9/2024, 01:40)

Hi Ana,

I want you to know that the Microchip business unit has not provided any actual solution. Based on the solution proposed for this support case I doubt that they have made any effort to understand what is at issue with my support request.

I want to make it clear that my initial requests and questions have never been solved.

I suspect that the issues you usually deal with are not this obscure. In short this is an issue with this specific controller failing to execute the machine code as documented in the data sheet.

It appears to me that Microchip does not want to document this fault or acknowledge it in any way.