TMS320 DSP
Number 29
DESIGNER'S NOTEBOOK
TEXAS INSTRUMENTS

Using the RBIT on the TMS320E25

Contributed by Keith Larson

Design Problem How does the TMS320E25 RBIT work?

Solution The RBIT primarily functions by disconnecting the internal program memory bus (PBUS) from the MUX which combines the internal data bus (DBUS) to create the externally shared program/data bus. The disconnect is made at the MUX and the internal nodes are left floating.

This diagram shows the location of the RBIT switch disconnecting the external and internal program spaces. The multiplier is shown as if it were receiving data as from a MAC instruction which will be discussed later.

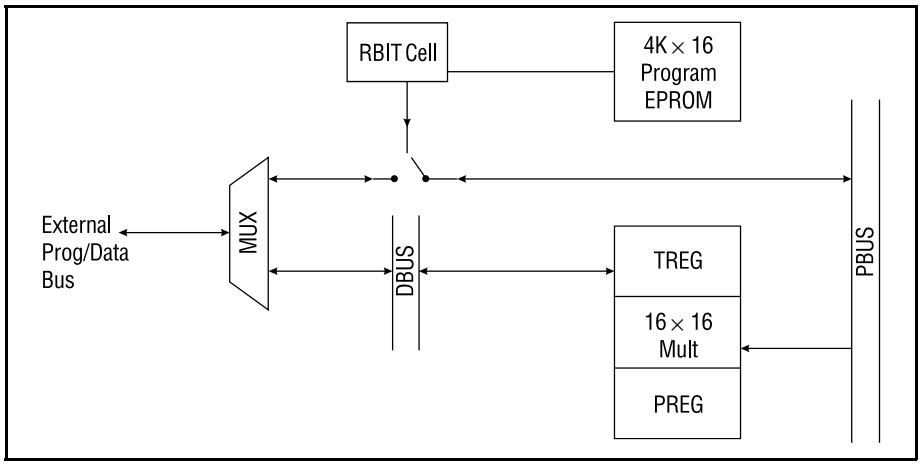

Figure 1.

What does this mean? On the TMS320E25 some instructions may appear to work and others will not. It all depends on whether or not an external data transfer from program or data space needs to be connected up to the internal program bus (PBUS). For instance, TBLW, BLKP, and other related mnemonics may appear to work when they are used to transfer external program memory to the internal data space connected to DBUS. You can probably quickly see that a transfer from the internal program space to the external data bus will not work. This also disallows any external code to be executed. This is what RBIT is supposed to do i.e., protect your code.

Other Points to Consider:

Note how the open switch disconnects the PBUS from the outside program space. This is why the µP mode will not work for a TMS320E25 after the RBIT is set. It also means that you cannot supplement your application with additional external code.

Secondly, the MAC instructions will not work with external program coefficients. In this case, the MAC instructions are supplying the on-chip multiplier with one operand from the DBUS and the other from the PBUS. The problem is that the external program space needs to be connected to the PBUS and the RBIT switch is in the way. To solve the problem, the coefficients should be moved to internal program RAM block B0 or read directly from the EPROM.

The RBIT also disables the EPROM programming mode, essentially disallowing an external EPROM programmer from reading the EPROM contents. It is therefore impossible to verify the EPROM contents once the RBIT has been set.

On the TMS320E1x devices the RBIT works by logically disabling the µC/µP pin and the EPROM programming mode. On the TMS320E25 this would not have worked since any opcode fetch from beyond the 4K boundary would constitute a breach of security. That is, a simple branch to an external debug routine would be all that is needed to get to the internal EPROM code. On the TMS320E1x devices, the entire program has to be on chip so nothing extra needs to be done.

Conclusion

The RBIT is a code integrity and security feature. Using a TMS320E25 with the RBIT set requires familiarity with the rules cited above.