# Rotary Motion Program Buffers

The rotary motion program buffers allow for the downloading of program lines during the execution of the program and for the overwriting of already executed program lines. This permits continuous execution of programs larger than PMAC's memory space, and also real-time downloading of program lines.

## Defining a Rotary Buffer

Each coordinate system can have a rotary program buffer. To create a rotary buffer for a coordinate system, address that coordinate system (**&n**) and send the **DEFINE ROT {constant}** command where **{constant}** is the size of the buffer in memory words. Each value in a program (e.g. **X1250**) takes one word of memory. The buffer should be sized to allow enough room for the distance ahead of the execution point you wish to load. Since most applications utilizing rotary buffers will not strain PMAC's memory requirements, it is a good idea to oversize the buffer by a good margin.

For instance, to be able to load 100 program lines ahead of the execution point in a four-axis application where using constant values for position (e.g. **X1000 Y1200 Z1400 A1600**), there should be at least 400 words of memory in the buffer, so it would be a good idea to allot 500 or 600 words for the rotary buffer (e.g. **DEFINE ROT 600**).

### Required Buffer State for Defining

In order for PMAC to be able to reserve room for the rotary buffer, there can be no data-gathering buffer and no rotary program buffer for a higher-numbered coordinate system at the time of the **DEFINE ROT** command. Therefore, delete any data-gathering buffer first, and define the rotary buffers from high-numbered to low-numbered. For instance:

```
DELETE GATHER
&3 DEFINE ROT 200
&2 DEFINE ROT 1000
&1 DEFINE ROT 20
```

## Preparing to Run

To prepare to run a rotary program in a coordinate system, use the **B0** command (go to beginning of program zero – the rotary program) when addressing that coordinate system. This must be done when no buffers are open, or it will be interpreted as a B-axis command. Once prepared this way, the program is started with the **R** command. This command can be given with the buffer either open or closed. If the **R** command is given for an empty rotary buffer, the buffer will simply wait for a command to be given to it, and then execute that command immediately.

## Opening for Entry

The **OPEN ROT** command opens all of the rotary program buffers that have been defined. Program lines following this are sent to the buffer for the host-addressed coordinate system (**&n**). Most users of rotary program buffers will have only one coordinate system, so this will not be of concern to them, but it is possible to switch coordinate systems on the fly and use several rotary buffers at once.

It is important to realize that after the **OPEN ROT** command, PMAC is treating as many commands as possible as buffered commands, even if it is executing them immediately (some commands mean one thing as an on-line command, and another thing as a buffered command). For instance, an **I100** command is a request for a value of I-variable 100 when buffers are closed, but it is a command to do a full circle with a 100-unit radius when a motion program buffer is open (the I-value is the X-axis component of the radial vector; since no axis positions are given, they are all assumed to be the same as the starting point)!

## Staying Ahead of Executing Line

The key to the handling of a rotary program buffer is knowing how many lines ahead it is; that is, how many program lines have been loaded ahead of the program line that PMAC is executing. Typically it will load ahead until a certain number of lines ahead is reached, and then wait until the program catches up to within a smaller number of lines ahead. A real-time application may just work one line ahead of the executing line; an application doing periodic downloading of a huge file may get 1000 lines ahead, and then start again when the program has caught up to within 500 lines.

### PR Command

There are several ways of telling how far ahead it is. First is the **PR** (program remaining) command, which returns the number of lines ahead. This provides a very simple polling scheme, but one that is probably not good for tight real- time applications.

### BREQ Interrupt

For tightly coupled applications, there are hardware lines to handle the handshaking for the rotary buffer, and variables to control the transition points of the lines. The BREQ (Buffer Request) line goes high when the rotary buffer for the addressed coordinate system wants more program lines, and it goes low when it does not. This line is wired into PMAC-PC's programmable interrupt controller, so it can be used to generate an interrupt to the host PC. (See Using the PMAC-PC to Interrupt the Host PCof this manual.) The complement, BREQ/, is provided on the JPAN connector. In addition, there is a Buffer Full (BREQ/) status bit for each coordinate system.

**I17 Stops Interrupts**

Variable I17 controls how many lines ahead the host can load and still get BREQ true. If a program line is sent to a rotary buffer, BREQ is taken low, at least temporarily. If still less than I17 lines ahead of the executing line, BREQ is taken high again, which can generate an interrupt. If it is I17 or more lines ahead, BREQ is left low. When a rotary program buffer is entered with **OPEN ROT** or the addressed coordinate system is changed, BREQ is taken low, and then set high if the buffer is less than I17 lines ahead of the executing point.

**I16 Restarts Interrupts**

Variable I16 controls where BREQ gets set again as the executing program in the rotary buffer catches up to the last loaded lines. If after execution of a line, there are less than I16 lines ahead in the rotary buffer, BREQ is set high. This can be used to signal the host that more program lines need to be sent.

By using these two variables and the BREQ line for interrupts, an extremely fast and efficient system can be created for downloading programs in real time from the PC.

**If the Buffer Runs Out**

If the program calculation catches up with the load point of the rotary buffer, there is no error; program operation will suspend until more lines are entered into the rotary buffer. Technically, the program is still running; a **Q** or **A** command must be given to truly stop the program.

If PMAC is in segmentation mode (I13>0) and is executing the last line in the rotary buffer, as long as a new line is entered before the start of deceleration to stop, PMAC will blend into the new move without stopping.

## Closing and Deleting Buffers

The **CLOSE** command closes the rotary buffers just as it does for other types of buffers. Closing the rotary buffers does not affect the execution of the buffer programs; it just prevents new buffered commands from being entered into the buffers until they are reopened.

**DELETE ROT** erases the rotary buffer for the addressed coordinate system and de-allocates the memory that had been reserved for it.