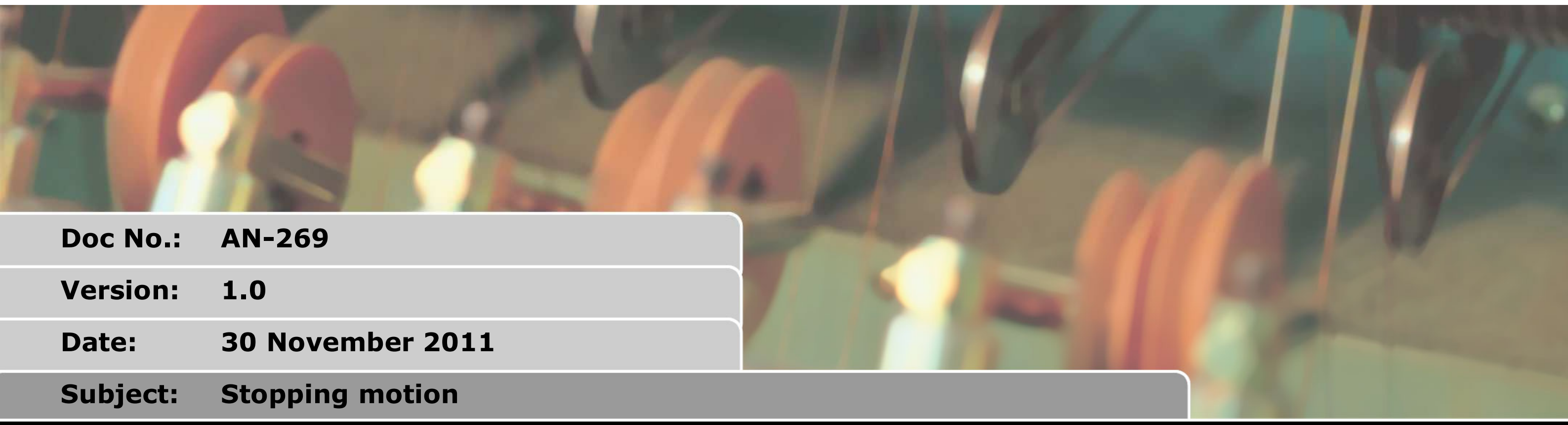
Trio Motion Technology Ltd.
Shannon Way, Tewkesbury,
Gloucestershire. GL20 8ND
United Kingdom
Tel: +44 (0)1684 292 333
Fax: +44 (0)1684 297 929

1000 Gamma Drive
Suite 206
Pittsburgh, PA 15238
United States of America
Tel: +1 412 968 9744
Fax: +1 412 968 9746

B1602 Tomson Centre
188 Zhang Yang Rd.,
Pudong New Area, Shanghai,
Postal code: 200122
CHINA
Tel/Fax: +86 21 587 97659



Doc No.: **AN-269**
Version: **1.0**
Date: **30 November 2011**
Subject: **Stopping motion**

APPLICATION NOTE

1. Introduction

Stopping motion can sometimes be more difficult than it sounds. You have to consider how many movements are buffered, how long your decelerations will take, if you want to stop one axis or multiple axes and more. This application note will cover most of the important considerations with examples of previous recommendations and current best practices.

2. Background to the problem

Usually a program will require some form of 'full stop' function which stops all motion on one or all axes. It is important to remember that the BASIC programs will issue motion to the motion buffers. So before any motion stop routine it is critical to stop any processes that load more motion.

Each BASIC program will be running in a PROCESS. Each process has a PMOVE buffer. This is where the motion command is first loaded. The PMOVE buffer checks which axes are involved with the motion command and then assigns the command to the correct axis motion buffers. If the motion buffers are full then the motion command will sit in the PMOVE buffer until the axis buffer has a space.

If the PMOVE buffer is full then the BASIC program will wait on the motion command until it can be loaded into the PMOVE.

Each axis has the MTYPE, NTYPE which are the currently running motion command and the first buffered motion command. By default LIMIT_BUFFERED =1 which means that there is only one buffered movement but it can be set higher to buffer more movements.

The full buffer system is shown in Figure 1, the NTYPE is buffer 1 and so any more buffers enabled by LIMIT_BUFFERED start at buffer 2.

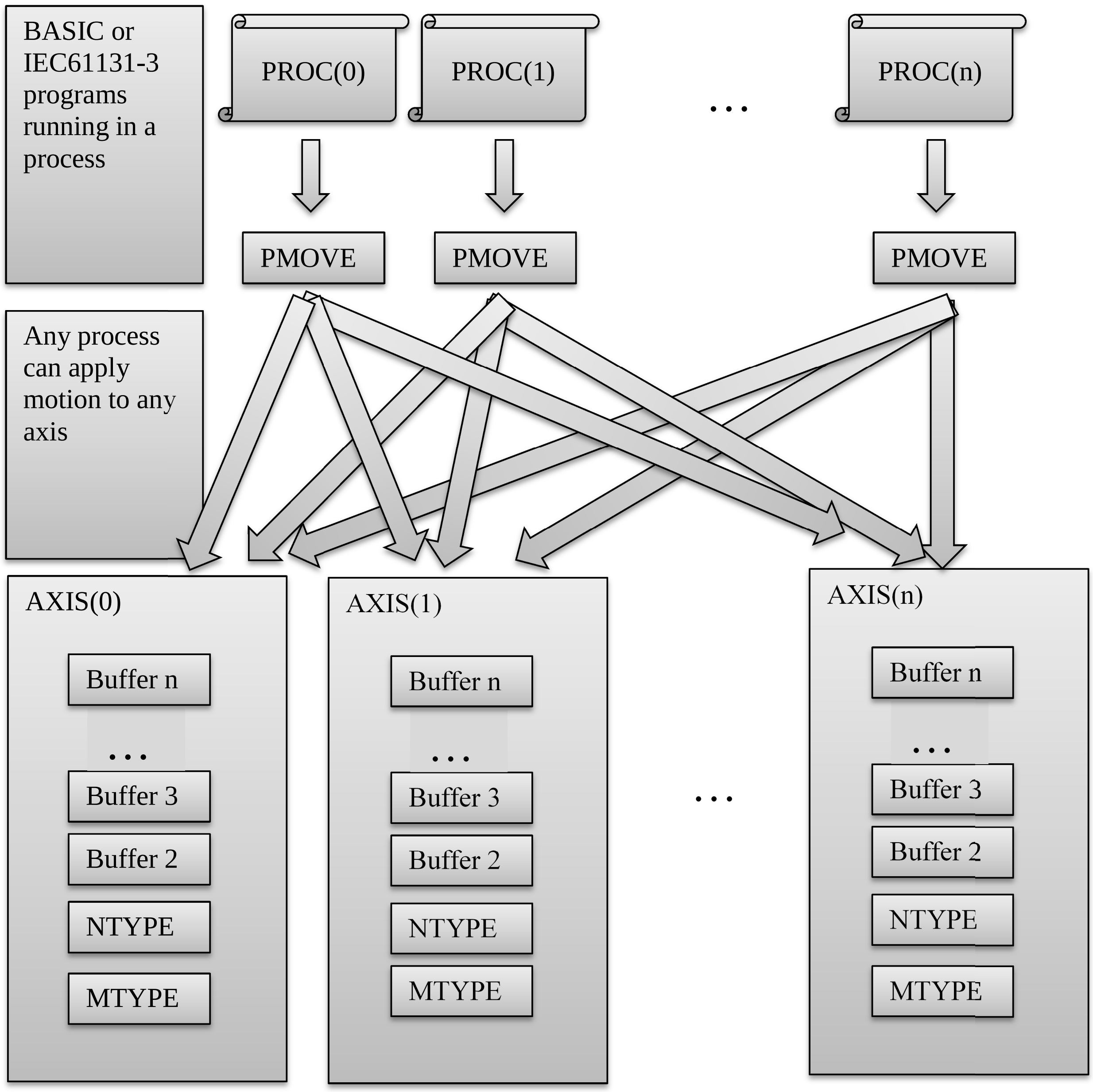


Figure 1: Buffer system

3. One axis vs Multiple axes

The techniques described below for stopping motion can be applied to a single axis or all axes. The difference is simply which command you use. CANCEL is to clear motion on one axis, RAPIDSTOP is used to clear motion on all axes.

This application note will now concentrate on cancelling motion on all axes, though you can simply change RAPIDSTOP for CANCEL to stop just one axis.

More information on CANCEL and RAPIDSTOP can of course be found in the Technical Reference Manual.

4. Traditional method

New features now make the following example redundant but it is important to understand its limitations in case you find this method used in an existing machine.

4.1. Code:

```
STOP "motion1"
RAPIDSTOP
WA(2)
RAPIDSTOP
WA(2)
RAPIDSTOP
```

4.2. Explanation

The first step is to stop any program that is generating motion.

The second step is to use the standard RAPIDSTOP function to clear the movement in the MTYPE. The WA's allow movements to load from the MTYPE and PMOVE buffers.

The first problem with this method is that firstly you need as many RAPIDSTOPS as you have buffered moves which currently can be up to 64. The second and bigger problem is that when you issue the RAPIDSTOP the motion command in the MTYPE will be CANCELled. If it is a profiled movement then it will decelerate at DECEL (or FAST_DEC). If this takes longer than 2ms then the next RAPIDSTOP will be issued against the movement that is currently cancelling.

When using this type of full stop routine it is assumed that the WA time is sufficient for a movement to decelerate and stop.

5. Clearing the buffered motion commands

This routine allows you to clear buffered movements independently of stopping the motion. It uses mode 1 of RAPIDSTOP or CANCEL which clears all buffered moves, but not the PMOVE.

5.1. Code:

```
STOP "motion1"
RAPIDSTOP(1)
WA(1)
RAPIDSTOP(1)
```

5.2. Explanation

As with the previous example it is important to stop any program that is generating motion.

The first RAPIDSTOP(1) will clear all buffered moves which include the NTYPE and other buffers enabled by LIMIT_BUFFERED. Any movement in the PMOVE will then be loaded into a buffer, which is cleared by the second RAPIDSTOP(1). Finally a RAPIDSTOP is issued to stop the currently executing motion command.

The WA is just to allow the PMOVE buffer to load into the buffered movements.

This routine does not stop the currently executing motion command, you could simply add another RAPIDSTOP to the end or just wait for the movement to naturally finish. Though if you have a continuous movement it will not end.

6. Clearing everything

This final routine will clear all active and buffered movements in one command.

6.1. Code:

```
STOP "motion1"
RAPIDSTOP(2)
```

6.2. Explanation

As with the previous examples it is important to stop any program that is generating motion.

The RAPIDSTOP(2) command will clear the PMOVE, all buffered motion commands and start a CANCEL on any active motion commands. This is the simplest and safest method of stopping all motion.

7. Other considerations

Some commands that can cause motion are not loaded into the motion buffers and so are not cleared by RAPIDTOP or CANCEL.

7.1. ADDAX

ADDAX links an axis to another. When you stop all motion it will not generate any movement but it will still be active. As part of your full stop routine you may wish to perform ADDAX(-1) to break the ADDAX link.

7.2. SYNC

SYNC is similar to ADDAX in that it links the BASE axis to a master but it also can apply synchronisation movements. It is recommended to add a SYNC(4) on any axis that uses SYNC as part of a full stop routine.