

Print Memory Utility

Page application

Mon, May 9, 2011

It is often of interest to know some bit of information from an entire set of nodes. The information desired may be located at the same address in memory in all nodes in the set. The usual Memory Dump page application can be used, of course, but typing node#s one by one can be tedious. The Print Memory page application can make this process easier. Optional features allow special access related to local application programs in a family of nodes, including access to the LATBL entries that include the LA parameters and access to the private variables of instances of a selected LA. A recent addition to this PA allows printing the most recent records of a data stream.

Page Layout

```
H PRINT MEMORY      06/13/97 1634
FILE<NBRF>         LIST<0576>
ADDR<00000E00>W
SIZE<  16>
LINE<  32>
NODE=06CF
CNTR=   21
```

The `FILE` parameter is a data file name. Here, the file name `NBRF` refers to a local file called `DATANBRF`—the file type name is always assumed to be `DATA`—that contains a list of all the `IRM` nodes in the Booster `HLRF` control system. This file can easily be created using the `MPW` assembler and linker and the `MPW` tool called `TFTPtool` to download the resultant `CODE` resource in the usual way. *If a single node# is entered rather than a file name, a temporary group is assumed that consists only of that single node.*

The `LIST` parameter is the node# whose serial port is used to target the lines of print output. Typically, this node would have its serial port connected to a terminal emulator program on a Macintosh or PC so the text can be captured for printing.

The `ADDR` parameter is the base address of memory whose contents are to be printed for each node in the node list file. The letter `B`, `w`, or `L` that follows the address means that the memory data is to be accessed by bytes, words, or longwords, respectively.

The `SIZE` parameter is the amount of memory starting at the base address that is to be printed, in the range 1–256 bytes. The `LINE` parameter specifies the format of how many bytes are to be printed on each line, in the range 1–32 bytes.

The `NODE` field and `CNTR` fields show the current node# and count of nodes that have already been processed during memory data acquisition as each node is queried every 15 Hz (or 10 Hz) cycle. Provision is made for nodes that deliver data late, say because they reside overseas, as well as for retrying a couple of times in case no response is forthcoming for the current node being queried. These two fields are output fields. After all the data has been collected and printed from all the nodes in the list of nodes, the `NODE` field will match the last node# in the list from which data was collected, and the `CNTR` field will show the total number of nodes from which data was collected.

Printed data format

```

FILE<NBRF> 06/13/97 1625
06B1:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06B2:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06B3:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06B4:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06B5:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06B6:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06B7:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06B9:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06BA:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06BB:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06BC:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06BD:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06BE:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06BF:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06C0:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06C1:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06C2:00000E00 0030 0048 0005 165D FFF5 FF3D 000E F3B6
06CA:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06CB:00000E00 0040 0048 0004 E827 7300 7F80 000E F896
06CE:00000E00 0010 0048 0000 0001 FFFF FC10 000E BCB8
06CF:00000E00 0030 0048 0005 0025 7300 7F80 000E F896

```

The above listing is an example of the printed output result of operating the program with the set of parameters shown. For each node# represented in the data file, the node# is shown followed by the memory address. After space for an error status indicator, if any, memory data is displayed in a format that reflects the mode of access. In this case the access was by words, so the data is shown as 16-bit words. If byte access were specified, the data would be shown as separate 8-bit bytes. If long word access were specified, the data would be shown as 32-bit longwords. Values that may be shown to reflect error status are likely to be only these two:

- 4 Bus error detected during memory data access
- 8 No response received from the node

Additional error codes 1-3, 5-7, indicate internal errors that should not occur.

Optional features for local applications

Each local application, when initialized for execution, allocates a block of dynamic memory for use in maintaining its context. The execution model is that a local application is invoked as a procedure every 15 Hz (or 10 Hz) cycle. It is passed an argument structure that includes the address of that allocated block, plus a set of parameters that the user specifies to particularize its operation. In this way it "remembers" what it was doing during previous invocations and the key operating values that influence its behavior. When a local application is disabled, if ever, its termination code releases this memory block. It is also possible to have more than one instance of a given local application running in the same node. For example, a closed loop local application might be used to regulate several similar devices, with a different set of parameters specified for each instance.

For diagnostic purposes, it is convenient to be able to list out some relevant fields of the context structure of the instances of a specified LA. This may be particularly helpful for the programmer who wrote the LA, as it requires some knowledge of the context block layout. To support this option for each node in a family, the program first accesses the target node to find out how many LATBL entries there are, then accesses the relevant contents of the node's LATBL, then searches for matches on the selected LA name, then uses the address to the allocated context blocks, applying a selected offset value, to access and list out the memory.

To specify the access to LA context fields, enter in place of the memory address the 4-character LA name, followed by a + sign, followed by 3 (hex) digits of offset, allowing for up to a 4K-byte offset. For example, <REQM+008> would indicate that all instances of the LA called REQM are to be found and memory listed beginning at 8 bytes offset from the start of that structure.

A similar option is also supported—that of listing a selected portion of the LATBL entries that correspond to the selected LA instances. This can be used to find out how many instances of a given LA are installed for each node of a family, as well as how the parameters compare across instances and/or nodes. The notation for this is similar to the previous option, but instead of a + sign, enter a –sign. For example, use of the notation <REQM-008> will list out memory from all LATBL entries that are in use with the selected LA, which with 24 bytes would show the address of the context memory, followed by the 10 parameter values.

Data stream record option

A recent feature that allows printing the most recent data stream records is specified by using the 8-character data stream name in the address field, such as <RETDL0G >. (This will not work so well for a data stream whose name looks like a valid hex address!) This option is useful because many data streams use records that are reasonably viewed in hex. For example, the time-of-day is normally recorded in BCD, which makes it easy to read when printed in hex.