

# Keeping up With the Unit Control Block: Part II

BY SAM GOLOB

It's hard to believe that this column is entering its eighth year. Time flies when you're having fun! This month, I'll continue last month's topic, which is an attempt to keep up with IBM's changes concerning the UCB, or Unit Control Block. As discussed last month, the UCB is required to define an I/O device to the MVS system. The UCB contains almost all of the information necessary for the device to be used in performing I/O requests. (A few device characteristics are stored elsewhere, but they are of minimal value.) It also records the status of a physical I/O device as represented by a sub-channel. With the advent of dynamic device reconfiguration, internals of the UCB

arrangement in storage have had to change, and UCB searching methods have had to be modified. Therefore, user programs and exits which access UCBs need to keep up with this restructuring. Many of these programs have stopped functioning properly in the new environment, and they need to be fixed.

One good place to find information from IBM manuals, especially concerning matters that we dealt with last month, is Chapter 23 of the *MVS Authorized Assembler Programming Guide*, which addresses "Accessing UCBs." Another is the *Planning, Installation, and Migration* manual for MVS Version 5. My intent isn't to rehash the information in these manuals, but rather to provide

an increased understanding of some of the internal processes involved. The ultimate result will be that you'll be better able to fix your own programs and get a deeper view of the entire situation.

IBM software planners have provided us macro interfacing methods to use in this new UCB environment. We now have to invoke certain IBM Assembler macros such as UCBLOOK, UCBSCAN, IOCINFO, and EDTINFO. No longer can we start at the beginning of the big block of UCBs in the nucleus area and chain through them in order. Instead, we must use this collection of macro interfaces because the system can now add and subtract UCBs dynamically.

**Figure 1: UCB Lookup Table**

This is a glimpse at the beginning of the UCB Lookup Table in storage, using the LOOK TSO command, which is a core browser. The LOOK command can be found on the CBT MVS Utilities Tape, in File 261. This diagram shows the ULUT header, as well as the beginning of the table, which is located at address 0204541C. The table itself consists of 12-byte entries, the last fullword of which is the actual UCB address. For clarity, I've marked the beginning of each table entry with a comma. This table is accessible to non-authorized programs, and can be used to scan UCBs, obtaining the actual UCB Common Segment address and not a copy of the UCB. This is better than the UCBSCAN service, but it isn't a documented interface. The last entry in the table is marked by a halfword of zeroes in the sequence number field. For a description of record layouts in this picture, see Figure 2. To ensure validity, you have to make sure that the IO Configuration has not changed. For this you may use the IOCINFO macro from SYS1.MACLIB, with the IOCTOKEN keyword.

LOOK COMMAND - DISPLAY VIRTUAL MEMORY  
ENTER CMD -  
LAST CMD - J+8

DISPLAY ASID= 00BC

02045398	>E4D3E4E3	01F50000	00000000	0204541C	*->ULUT. 5. .... *
020453A8	000009F1	000000B8	0000002C	000004D0	*... 1. .... *
020453B8	00000280	0000000D	00000000	000001B0	*..... *
020453C8	02190007	04F200E7	00010000	00270000	*..... 2. X. .... *
020453D8	00003BA0	00001DD0	00000EE8	00000774	*..... Y. .... *
020453E8	000003C0	000001E0	000000F0	00000078	*..... 0. .... *
020453F8	0000003C	00000018	0000000C	0000000C	*..... *
02045408	00000000	00000000	00000000	00000000	*..... *
02045418	00000000	, 00014000	00020000	00EF2168	*..... *
02045428	00024000	00030000	00EF21D8	, 00034000	*.. ..... Q. .... *
02045438	00040000	00EF2248	, 00044000	00050000	*..... *
02045448	00EF22B8	, 00054000	00060000	00EF2328	*..... *
02045458	, 00064000	02120000	00EF2398	, 00104000	*..... *
02045468	00080000	00EF2408	, 00114000	00090000	*..... *
02045478	00EF2478	, 00124000	000A0000	00EF24E8	*..... Y* .. *
02045488	, 00134000	000B0000	00EF2558	, 00144000	*..... *

1= HELP  
7= BACKWARD

2=  
8= FORWARD

3= END  
9= HIST BWD

4=  
10= HIST FWD

5= REPEAT  
11=

6=  
12=

**Figure 2: Accessing the UCB Lookup Table**

This figure shows you how to access the UCB Lookup Table in storage, and describes the ULUT header and the table entries. This description is courtesy of Gilbert Saint-flour, and comes from the source code of his SHOWMVS TSO command.

```
* - - Pointer from IOCOM to IOVT - - - - -
IOCI OVTP EQU IOCOM+X'0D0', 4, C'A' V(IOVT)
*
* - - Description of the relevant parts of the IOVT - - -
IOVT      DSECT      IOS VECTOR TABLE      ESA41
          DS         C' IOVT'
          DS         H' 384'      LENGTH OF IOVT
          DS         XL2
IOVTULUT  DS         V(ULUT)      UCB LOOK-UP TABLE
          DS         3F
IOVTCDA   DS         V(CDA)      CONFIG DATA AREA
*
* - - X' 84' Bytes - - - - - Description of ULUT Header - - -
ULUT      DSECT      UCB LOOK-UP TABLE      ESA41
          DS         C' ULUT'
          DS         X' 01F5'      ?
          DS         XL2, XL4      UNUSED
ULUTENTP  DS         A(ULUENTRY)  FIRST LOOK-UP ENTRY
ULUTENTN  DS         F            TOTAL NUMBER OF LOOK-UP ENTRIES
ULUTTAPE  DS         F            NUMBER OF TAPE LOOK-UP ENTRIES
ULUTCOMM  DS         F            NUMBER OF COMM LOOK-UP ENTRIES
ULUTDASD  DS         F            NUMBER OF DASD LOOK-UP ENTRIES
ULUTDISP  DS         F            NUMBER OF DISP LOOK-UP ENTRIES
ULUTUREC  DS         F            NUMBER OF UREC LOOK-UP ENTRIES
ULUTCHAR  DS         F            NUMBER OF CHAR LOOK-UP ENTRIES
ULUTCTCA  DS         F            NUMBER OF CTCA LOOK-UP ENTRIES
          ORG         ULUT+132
* - - 12 Bytes per entry - Description of ULUT Entries - - -
ULUENTRY  DSECT      UCB LOOK-UP ENTRY
ULUEDEVN  DS         H            DEVICE NUMBER
ULUEFLGS  DS         X' 4000'     FLAGS
ULUEDYN   EQU        X' 40'      DYNAMIC UCB
ULUESEQN  DS         X' 0001', XL2 SEQNO
ULUEUCBP  DS         V(UCBOB)    UCB ADDRESS
ULUELEN   EQU        *-ULUENTRY
*
```

A valid UCB can actually disappear. Yet, all of the currently valid UCBs must still be searchable. These macros are now the only supported IBM search method.

There is another fact to note. UCBs have “segments” and “extensions” of various types. The main part of the UCB is called the “UCB Common Segment.” The other segments and “UCB extensions” can be reached in various ways, once we have access to the UCB Common Segment. Looking at the macro IEFUCBOB from SYS1.MACLIB will give you specific help. Our discussion will concern how to access the UCB Common Segment only. In the past, other segments were mostly contiguous in storage with the UCB Common Segment. Today, the other segments and extensions usually are nowhere near the location of the Common Segment. This is another reason why our search methods have to be different.

With the help of my friend Gilbert Saint-flour, I’ve discovered a few of the undocumented internals of how UCB searches now

work. I cannot recommend their direct use, because IBM reserves the right to change the interface. Nevertheless, knowledge of these mechanisms is instructive and illuminating, and we can profit from seeing more of how the UCB searches are actually happening.

### ***THE UCB LOOKUP TABLE (ULUT)***

Back in the MVS/370 days, UCBs were scanned using a “lookup table.” This was a sequentially read table which (eventually) pointed to all of the UCBs in the system as you went through it. When MVS/XA arrived, completely changing the “channel, path, device” scheme of device addressing, the proper UCB scanning method became a call to a routine, pointed to by an address in the CVT, CVTUCBSC. This method might still work for “static” UCBs, but its effectiveness is no longer to be relied upon.

Now, to implement dynamic UCBs, IBM has again reverted to a modified version of the “lookup table” arrangement. IBM has hidden this arrangement from public view,

calling it through the use of the UCBSCAN macro. But, the internals of a UCB lookup are built on this table. Every time a dynamic device reconfiguration is done, the UCB Lookup Table (ULUT) gets rebuilt.

On MVS/ESA systems (from release 4.1 through at least release 5.2), the ULUT can be found in the following way. From the CVT (Communications Vector Table), which is the anchor for MVS control blocks, you can look at the contents of field CVTIXAVL at X’7C’ from the beginning of the CVT. This field points to the address of the IO Supervisor’s Communication Area (IOCOM). The address at displacement X’D0’ from the beginning of the IOCOM points to the IOS Vector Table, or IOVT. Finally, the address at 8 bytes from the beginning of the IOVT points to the ULUT.

You can use a core-browsing program, such as LOOK, from File 261 of the public domain CBT MVS Utilities Tape, to actually see what this storage looks like. The CBT Tape, with its many tools, can be obtained through NaSPA. See Figure 1 for a “LOOK” at the beginning of the ULUT.

The actual ULUT has two parts: its header section, and the actual UCB lookup table entries. See Figure 2 for an attempt at a DSECT description of the ULUT. Remember that this is an undocumented interface.

Gilbert Saint-flour, who wrote the encyclopedic SHOWMVS TSO command (on File 183 of the CBT MVS Tape), uncovered many details of the ULUT’s structure. He has written actual code to use this interface in his SHOWMVS command. Part of the “UCB” output from his code is shown in Figure 3.

SHOWMVS was designed to display, under ISPF or in batch, as much information about your MVS system as possible. As such, SHOWMVS is an excellent auditor or systems programmer tool. Besides its some 30 other displays, its UCB display shows all online devices in each device class, who has a dataset most recently allocated on the device, and SMS information (if the device is SMS-managed).

SHOWMVS, while running under ISPF, will initiate a subtask that constantly tries to keep all the displays up-to-date. Because of this, for the UCB display, actual UCB addresses are needed to obtain current information. UCB copies will not do. Therefore, Saint-flour couldn’t employ the standard UCBSCAN interface, which for unauthorized programs, will return only a copy of the UCB and not the real one. He needed to supply the address of the real UCB in an unauthorized environment.

The ULUT 12-byte table entries may be read sequentially by any program. They consist of a halfword hex device address,

a halfword of flags, where the X'4000' bit (indicating a dynamic UCB) is the only one currently used, a halfword sequence number, and an unused halfword. The real UCB common segment address follows. The last entry of the table is marked by zeroes in the sequence number field. Or, it can be calculated by "length" information found in the ULUT header. Saint-flour employs the latter method in SHOWMVS.

I hope this technical glimpse will enlighten you and arouse your curiosity. SHOWMVS source code, including all of the methods it uses to obtain system information, is public-domain. You can learn a lot from it. This code may be found on File 183 of the CBT MVS Tape. Now we'll finish our discussion with some program conversion hints.

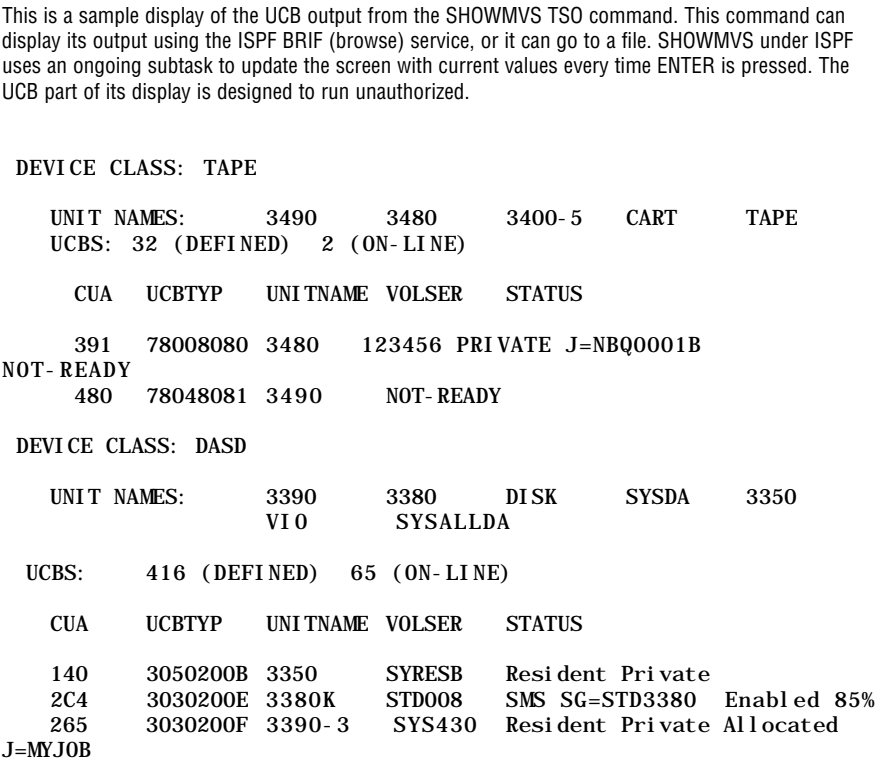
**FIXING EXISTING USER PROGRAMS**

IBM has classified UCB search methods into two classes: limited methods and general methods. The idea is that the "limited" methods cannot be used to find dynamic UCBs, just statically defined UCBs. The "general" methods, which are the newer ones, supposedly can be used to find all defined UCBs, optionally including 4-byte unit addresses and dynamically defined UCBs. I might add that there are some older UCB finding methods which no longer work at all.

The limited methods mentioned in Chapter 23 of the *Authorized Assembler Programming Guide* are the IOSLOOK macro for finding a single UCB, and the UCB scan SERVICE for scanning multiple UCBs. The UCB scan SERVICE is reached from the CVT at displacement X'434'. This points to the entry point address of the routine IOSVSUCB. Programs using these methods (you can see what they're doing by looking at their source code) should be converted to use the UCBLOOK macro to find single UCBs, and the UCBSCAN MACRO to do a comprehensive search. As I said before, the UCBSCAN MACRO has a limitation for non-authorized programs. It will return a copy of the UCB only, and not the address of the actual UCB.

For programs which find units by generic or esoteric unit names (see "MVS/SP System Modifications," Chapter 6), the old way (XA and above) was to use the Unit Verification Service, module IEFEB4UV (which has 11 different function calls). IBM has provided the macro EDTINFO to supposedly replace this service, but the function calls in the EDTINFO macro don't exactly match the old calls that were possible with the IEFEB4UV routine. A certain amount of ingenuity will be necessary for some of the calls if EDTINFO is to obtain the same result.

**Figure 3: UCB Output From the SHOWMVS TSO Command**




An example of a program needing this kind of conversion is the UNITS command from File 360 of the CBT MVS Tape. UNITS is a TSO command which will either return all defined generic and esoteric unit names alone (this part still works), or will also return all the unit addresses together with each generic name. The latter function doesn't work properly anymore, because IEFEB4UV cannot find any dynamic UCBs. IEFEB4UV comes up empty when asked to display a list of unit addresses for a given unit name, if they were all defined as dynamic.

I haven't fixed the UNITS command yet. When I do, I'll try and share it with you. If you fix something, please send it to the CBT Tape submission address so you can share with everyone too. The address may be obtained from NaSPA, 7044 S. 13th St., Oak Creek, WI 53154 (414) 768-8000. That's how this work gets divided. Everyone benefits because no one individual has to do too much of the total job.

I'd like to close with a general comment about IBM publications. You'll need them to get more help on this and other topics. Being one of the largest publishers in the world, IBM would much rather sell you manuals on CD-ROM than on paper. If you have to research a certain topic, such as UCB lookups, it's much easier to do so when you have all the manuals at hand on your PC.

The MVS Omnibus Collection, available through your IBM representative, has all of the MVS manuals you'll ever need. The Rainbow Collection has all the "Red Books" and "Yellow Books" etc., from all of the systems centers. With them, you'll be better able to deal with any problem.

I hope this column has been enlightening. I'll see you next month. .

**Sam Golob is a senior systems programmer working in New York City.**

©1996 Technical Enterprises, Inc. Reprinted with permission of **Technical Support** magazine. For subscription information, email [mbrship@naspa.net](mailto:mbrship@naspa.net) or call 414-768-8000, Ext. 116.