

**LOGIC ANALYZER USING SOURCE PROGRAM  
OR OTHER USER DEFINED SYMBOLS IN THE  
TRACE SPECIFICATION AND THE TRACE  
LISTING**

**CROSS REFERENCE TO RELATED  
APPLICATION**

This application is a continuation of application Ser. No. 481,010, filed Mar. 31, 1983, now abandoned.

**FIELD OF THE INVENTION**

The subject matter of the present invention pertains to improvements in logic state analysis, and may apply to both logic state analyzers and the debuggers used in conjunction with emulators.

**BACKGROUND AND SUMMARY OF THE  
INVENTION**

The development of executable code for processor based systems frequently involves the use of compilers and assemblers that produce relocatable object code. When such programs are subsequently linked and loaded a trace listing provided by a logic state analyzer can be at best tedious to appreciate, and at worst extremely difficult to follow. Even a reverse assembler can not replace a reference to an arbitrary address with the corresponding symbol used in the original source program. To do that requires an appreciation on the part of the user of how the various software tools interact, and of how they modify the relocatable object code to produce a final absolute value. A considerable amount of tedious nondecimal arithmetic may be required to relate the actual events reflected in the trace to a collection of source program listings. The situation would be bad enough in cases where the hardware in the target system is known to be good, and what is being debugged is simply the software. But in many development situations there may be bugs in both the hardware and the software. This makes it especially important to be able to rely on the trace for information about what really happened, as there may well be a discrepancy between actual events and the legitimate aims of even a properly written program. Under these circumstances it would be less wise to think of the trace as a hardware version of the program listing, and more useful to think of the program listing as a guide to understanding the trace. In these types of situations the extra overhead of "unrelocating" a trace can be particularly burdensome. It would be especially desirable if all absolute values for addresses and operands in the trace were replaced with notations involving the symbols used by the programmer in the original source programs. Such symbols might refer to individual locations or to ranges of locations. It would be useful if similar symbols could be defined in addition to the ones found in the source listings, without having to edit the sources to include them. It would also be helpful if references to original source program line numbers could be included in the trace, or even actual source lines. This would aid a great deal in allowing the user to follow overall program flow.

Another development or troubleshooting situation pertinent to the invention can arise in connection with the operation of finite state machines. A trace of such a state machine is a sequential series of states: e.g., 001001, 010001, 010011, etc. It is frequently the case these states can be given labels, such as "INC\_P\_REG", (incre-

ment P register), "WAIT\_MEMC", (wait for Memory Complete), or STM (Start Memory Cycle). It would be desirable if a logic analyzer could provide a listing of the trace in terms of such labeled states. Each state in the listing would either be a label or a value relative to a label. In the latter case there might be several states in some process, say a read memory cycle. The label RMCY might refer to the first state in the process. RMCY + 3 would denote the fourth state therein without having to invent labels for every separate state in the process (and by implication, in the entire machine).

The ability of the logic state analyzer described herein to integrate source program symbols and source lines into the trace listing arises from giving that analyzer access to the symbol tables produced by any compiler or assembler that produced the code (whether absolute or relocatable) and by giving the analyzer access to the decisions made by the linker or relocating loader. Using this information the analyzer can determine by various inspection processes what symbols to use in the trace listing.

A further benefit emerges from the ability to do this. It is then also possible to at once expand and simplify the process of defining the trace specification for the analyzer. The trace specification tells the analyzer under what conditions to commence the trace and exactly what type of information to include therein. With the aid of the invention it is possible to use source program symbols in the trace specification without having to learn what their absolute values are at run time. This is quite useful, as those absolute values are apt to change as bugs are found and fixed, or as different versions of the software are developed and tested. But an analyzer constructed to take advantage of the various symbol tables and the load map need not have its trace specification altered merely because one or more programs are of different lengths than before, or because the programs are loaded in a different order. The symbolic nature of a "relocatable trace specification" makes that unnecessary.

These principles may be extended to apply to logic state analysis performed upon target systems that incorporate memory management units. In such target systems the relocated addresses issued by the processor are virtual addresses that are further modified in real time by the memory management unit as the processor runs. The modified addresses are the actual physical addresses sent to the memory. Their values are contingent upon run time conditions reflecting what parts of memory are allocated to which programs or tasks. These allocations are dynamic, and generally cannot be given in advance as fixed absolute offsets to be applied to the relocated addresses present at run time. Those relocated address are themselves offset by some relocation base from the relocatable addresses issued by the assembler or compiler, as mentioned above. Thus, such a memory managed address involves some absolute value that results from dynamically offsetting a relocated value that is already offset by a fixed amount from the relocatable value appearing in the source listings.

The dynamic offsets mentioned above need not be entirely private to processes within the target system, and therefore mysterious to the logic state analyzer. The symbols representing the various dynamic offsets can be defined to the logic state analyzer. Then provided certain criteria pertaining to keeping public knowledge of the offsets current (a task specific to the