Copies: F. Corbato
R. Decker
D. Edwards
T. Glazer
J. Gildersleeve
H. Lee
A. Longanecker
H. Mathews
D. O'connor
J. Saltzer
R. Ward
J. Wilde
E. Vance

TO: W.L. Estfan

SUBJECT: Multics Diagnostic System Trip Report

DURATION AND LOCATION OF TRIP: Monday, May9 to Friday May 27 at MIT, Cambridge, Massachusetts

OUTLINE:

Harlow Frick
645 System Design Unit
June 2, 1966

1.0 INTRODUCTION

The purpose of this document is to inform you of progress and direction taken on the definition of an on-line diagnostic system running under Multics control. Also, to make recommendations and solicit constructive comments and suggestions.

Many important problems have been overlooked or side-stepped in this document and some of the concepts and details described herein are expected to change. For this reason, it is important that distribution be limited so as not to propogate terms which are incorrectly used and some faulty concepts.

Recommendations as to which of our existing 600 line T&D programs should be modified to run in the Multics environment will be delayed until the diagnostic system needed for the Multics environment is further defined and until I have a better understanding of Multics.

Although many of our existing programs could probably be modified to run in the Multics environment, I suspect that it would be better, considering long range goals, to re-write them in EPL (Early Programming Language - a subset of PL1) or in 645 assembly language. A stronger position will be taken on this in the Multics Diagnostic System Position Paper which should be completed in about a month.

2.0 MULTICS/DIAGNOSTIC SYSTEM INTERFACE REQUIREMENTS

Existing Multics modules as described in the MSPM (Multics System Programmers Manual) leave ample room for adding a diagnostic system without many changes to what is already defined. As the diagnostic interface is defined, additions will be made to the presently defined modules and new modules will be specified.

The following comments about some of the presently defined Multics modules are intended to illustrate the extent to which additional details must be added in order to provide a proper diagnostic system interface.

<u>Traffic Controller</u> -	Some additional details are needed.
<u>Transactor</u> -	Considerable additional definition is required in this module. In this document I have delegated several responsibilities to the transactor for convenience only. It is likely that the Transactor will be divided into several different modules as various functions are further defined.

Device Control Modules (DCM) -	(also known as DIMs) - No change is contemplated to any of the DCMs. The diagnostic system will either use the DCMs as other users or will replace a DCM for a particular device when exercising the device with a diagnostic program.
GIOC Interface Module (GIM) -	No change is contemplated to the present definition of the GIM, but considerable additional detail is required. The GIM can be used as presently defined when exercising a device with a diagnostic program. Failures in the GIOC should be detected where practical by the GIM. When the GIM detects a GIOC failure, certain information about the failure should be stored and a recovery procedure should be initiated. This is a Big problem area in which lots of additional work is required.
MSU Interface Module -	I have not yet investigated this module.
Other Modules -	Not yet investigated.

3.0 SOFTWARE RECONFIGURATION

3.1 A method of allocating components to the diagnostic system.

The diagnostic system should have authority to test only those components assigned to it. The transfer of components to and from the diagnostic system should be handled by the transactor.

The following is a possible approach:

An unassigned component file (or possibly an "unassigned" bit in the Physical configuration file) defines all components which are physically connected to the system but which are not currently allocated for use by Multics. Only the Transactor can transfer components to and from the unassigned component file.

A diagnostic operator can run diagnostic programs on only those components which are in his system configuration file. Any number of diagnostic operators can be currently logged in. Any diagnostic operator can transfer components at will between his system configuration file and the unassigned component file, via requests to the transactor.

The Transactor transfers components between the Multics configuration file, the unassigned component file, and diagnostic system configuration files as follows:

1. Transfer from Multics configuration file to unassigned component file.

   Multics requests the transfer because the component has malfunctioned or is due for preventive maintenance, or because a Multics operator requested the transfer.

   A Multics operator may request the transfer for any of the following reasons:

   a) The component is to be put "off-line" for preventive or corrective maintenance.

   b) The component is to be physically removed from the system.

   c) A diagnostic program is to be run on the component to determine whether it functions properly.

2. Transfer from the unassigned component file to the Multics configuration file.

   This is done only by a request from a Multics operator.

3. Transfer from the unassigned component file to a diagnostic system configuration file.

   Any diagnostic operator may transfer components from the unassigned component file to his system configuration file.

4. Transfer from a diagnostic system configuration file to the unassigned component file.

   Any diagnostic operator may transfer components in his system configuration file to the unassigned component file.

5. Initially there will be no way of transferring directly between the Multics configuration file and a diagnostic system configuration file. The reason is to keep these transfers under control of the Multics operator.

3.2 A Simple Re-configuration Example

If a printer is to be moved from GIOC channel X to channel Y, and a card reader is to be inserted into the system at channel X, the procedure might be as follows:

a) Multics operator requests the Transactor to move the printer assigned to channel X into the unassigned component file at the next logical breakpoint.

b) At the next logical breakpoint, the transactor moves the printer from the Multics configuration file to the unassigned component and informs the Multics operator.

c) The printer cables are physically moved from channel X to channel Y. The card reader is physically moved into the area and its cables are connected to channel X.

d) The Multics operator orders the Transactor to delete printer on channel X from the unassigned component file, and add a card reader on channel X and a printer on channel Y.

e) A diagnostic operator orders the Transactor to transfer card reader on channel X and the printer on channel Y to his system configuration file.

f) Diagnostic operator runs a short card reader and printer diagnostic to verify that these components are operational.

g) Diagnostic operator orders the Transactor to give his system configuration back to Multics.

4.0 RECOMMENDATIONS

4.1 One or more diagnostic programs, which run in the Multics environment, should be provided for each component which may be deleted from the Multics system configuration without curtailing Multics operation. The component must be deleted from the Multics configuration and allocated to the diagnostic system while the diagnostic is being run. The following is an incomplete list of components which diagnostic programs should be provided for.

1. Entire <u>MSU</u> system
2. <u>One MSU sector</u>
3. <u>Sixteen contiguous MSU sectors</u>
4. Entire <u>Processor</u> system (plus one memory controller for diagnostic program use).
5. <u>Memory Controller</u>
6. Entire <u>GIOC</u> system
7. <u>GIOC Adapter</u>

   At least one program should be provided for each type of GIOC adapter.
8. <u>GIOC Device</u>

   At least one program should be provided for each type of device.

4.2 No general distinction should be made between diagnostic processes and other processes as far as loading, execution, or maintenance is concerned.

Poorly put

4.3 The Multics Monitor should be capable of starting diagnostic processes for any of the following reasons:

a) A diagnostic operator requests startup.

b) Elapsed time interval triggers startup.

c) System process detects a failure and therefore triggers startup.

4.4 Most diagnostic programs should be written in EPL and should run in slave mode. In fact, the only exceptions should be diagnostics for major modules when the entire major module is down. For example, in a multiple GIOC system, when one GIOC is completely down, a special diagnostic GIM (GIOC Interface Module) may replace the standard GIM when communicating with the GIOC which is down. The diagnostic GIM is not necessarily coded entirely in EPL and it may include some code written in Master Mode. (Note: Further investigation is required before we commit ourselves to EPL. However, we should attempt to use EPL unless there is a valid reason for not doing so. Also, ? any existing T&D programs modified to run in the Multics environment would, of course, be assembled with the 625/635 assembly program.)

4.5 Wherever practical the initial version of Multics should include internal software checks for verification of proper operation of the system. This will help isolate software as well as hardware failures. This approach may easily be overdone. The philosophy should be to assume that the processor works and protect against I/O system failures causing the software to fail. The following kinds of software checks should be included for the prototype system:

a) Verification by software, that GIOC status control words are properly maintained by the hardware and that a status word is indeed stored for each increment of the status control word.

Rather than force zero

b) If a status word is supposed to be stored by the hardware with zeros in a field, the software should verify that the field is indeed zeros before performing an operation (like entering a branch table) which would cause an undefined software failure if the hardware failed.

4.6 An additional person should begin working full time on the Multics diagnostic system as soon as possible. (I recommend Al Longanecker.) in Consulting? This person should investigate the feasibility of writing diagnostic system programs in EPL and should also write a Multics Diagnostic System Functional Specification.

5.0 DEFINITIONS

<u>component</u> -	A discrete physical portion of the system hardware which may be allocated to or deallocated from Multics use.
<u>diagnostic program</u> -	A program specifically written to help engineers diagnose hardware failures. There is no implication as to the amount or usefulness of error output information.
<u>diagnostic process</u> -	Basically, a diagnostic program in execution in the Multics environment.
<u>test program</u> - test process?	A program specifically written (or in some context, used) to determine whether a system or some part of a system is functioning properly.
<u>console user</u> -	A person communicating with the Multics system via an on-line typewriter like device.
~~fu~~ personal identification	Note: A console user's domain of authority is always defined by his user number, not necessarily by which physical device he is connected to.
<u>multics operator</u> - helpful?	A console user with approximately the same responsibility and authority as the operator in a GECOS environment.
<u>diagnostic operator</u> -	A console user whose user number defines him as having the following authority:

1. All authority of an ordinary console user.
2. Authority to use system components which have been de-allocated from the Multics configuration. In other words, he can transfer components between the unassigned component file and his System Configuration File.
3. Authority to execute Diagnostic Programs.
4. Authority to communicate certain information to and from the Multics system.

per dir.

<u>diagnostic system programmer</u> - A console user whose <u>user number</u> defines him as having all authority of a diagnostic operator plus the authority to change certain permanent system files. For simplicity I have defined two types of diagnostic system files.

<u>Type A diagnostic system files</u> may be improperly changed without resulting in a Multics system failure to anyone except diagnostic operators. These files contain most permanent diagnostic system symbolic and binary program files and various standard data files. Most diagnostic system programmers will have authority to change only type A files, and perhaps only those type A files to which individual responsibility is assigned.

[illegible]

<u>Type B diagnostic system files</u> contain symbolic and binary files of Multics Modules which are used primarily by the diagnostic system. They are implemented and maintained by diagnostic system programmers. Great care must be taken when changing some of these files because an improper change may cause catastrophic failure of the Multics system. It is likely that only certain on-site system programmers will have authority to change type B files.

6.0 PROJECTED WORK PLAN

Upon returning to MIT I will continue to spend considerable time familiarizing myself with the Multics system. I also plan to prepare the following documents.

1. <u>Multics Diagnostic System Position Paper</u>

   Planned completion on FW 27 (7/1/66)

   This document will explain the general approach to be taken on the Multics Diagnostic System. The following topics may be discussed:

   a. General role which diagnostic programs should play regarding error detection and correction versus the role of test programs and the rest of the Multics software.

   b. Feasibility of modifying the 645 off-line T&D system to run in the Multics environment.

   c. Languages which the diagnostic system should be written in.

d. Diagnostic system operating procedures.

e. System component allocation for diagnostic testing or maintenance.

f. System components which diagnostic programs should be provided for.

g. System re-configuration.

2. <u>Multics/Diagnostic System Interface Functional Specification</u>

Planned completion on FW 31 (7/29/66)

This document will describe in general, what changes or additions are required in Multics modules defined in the MSPM (Multics System Programmers Manual) in order to allow implementation of the Multics Diagnostic System described in the Multics Diagnostic System Position Paper.

3. <u>Multics/Diagnostic System Interface Design Specification</u>

Planned completion on FW 35 (8/26/66)

Same as 2 except in detail.

4. <u>Multics Users Guide for Diagnostic Programmers</u>

Planned completion on FW 39 (9/23/66)

This document will contain information which a diagnostic programmer will find useful when starting to implement a diagnostic program which runs in the Multics environment. It may include the following:

a) Guide for using Multics Documentation.

b) Guide for using programming languages.

c) Suggested Program Documentation Standards.