February 7, 1967

TO: MSPM Distribution

FROM: G.S. Stoller

SUBJ: BE.7.12

Segment ESCAPE is not yet storing associative memory and the DBR in the communication area, but this will be implemented soon.

Published: 2/7/67

Identification

Execution Environment for the 645 Pseudo-Process

V. B. Nguyen
D. E. Joel
D. H. Slosberg
G. S. Stoller

0. Introduction

6.36 and 64.5 jobs will be submitted and run on the GE-645 exactly as they were on the GE-635 except for the '645 pseudo-process' activity which will be run without simulation.

Each 635 activity on the GE-645 will run under a GECOS configured for one IOC. The GECOS Supplement will run one GIOC as an IOC for GECOS. Separate documentation will be provided for those 645 pseudo processes which have a need to handle I/O directly in this environment. This will be done through a second GIOC or by a single GIOC shared with GECOS.

Operating procedures

The main characteristics of the operation at load/execute time are as follows:

1. The GECOS Supplement, on request, yields information about the '635 Slave' activity currently in execution. This allows the 645 loader to determine its own environment.
2. The GECOS Supplement, on request, sets a '645 Pseudo-Process' switch which causes interrogation of a user's pseudo fault and interrupt vectors when a fault or appropriate interrupt occurs, and reflects the condition to the user as indicated by the contents of the appropriate vector.

3. When a fault condition occurs with the '645 Pseudo-Process' switch on, and the appropriate entry in the user's pseudo fault vector indicates that the user is not handling this condition, the GECOS Supplement goes to its '645 pseudo-process terminate' routine. This stores various hardware registers into a fixed communication zone (see figure 5) and passes control to a termination routine in the 635 escape coding through this communication zone. This permits the normal form of dump given by 6.36 or 64.5 to be obtained in preference to the GECOS abort dump.
4. It is normal practice to use two libraries to set up execution. One contains 635 subprograms (e.g., the 645 loader), and the other contains 645 assembled segments (especially the special inclusion segments). Any changes required converting from simulation to execution operation are handled entirely by manipulation of 635 subprogram library.
5. The 645 Pseudo-Process placed into execution is in a position to do anything it wants to do, and thus can cause a catastrophic software failure. It is a design objective that the user have this freedom while still trying to catch error situations if possible (see 3 above).

The specific requirements of the system are described below in some detail, with particular emphasis given to interface details.

1. The GECOS Supplement

When a 635 slave activity is set up, the basic mechanism is that of building a descriptor segment. The entries which are required in the descriptor segment are described in Figure 1.

Segment Number	Description	Descriptor Attributes
5	Describes the assigned 635 slave memory, upaged, 1024 word blocks.	Slave procedure, slave access, write permit.
6	Describes the mailbox area for the GIOC assigned to 645 pseudo-process work only, unpaged.	Data

Figure 1. Descriptor Segment for 635 Slave Activity.

It is necessary that the 645 loader make requests of the Supplement.

The vehicle used to make requests of the Supplement, is the Master Mode Entry 1 command. The specific requests which can be made are as follows:

a. Return a descriptor word.

```
LDA     1,DL
LDQ     N,DL
MME1    64
```

The N'th word in the descriptor segment (N starts at zero) is passed back by the Supplement in register A.

b. Change Mode, set '645 Pseudo-Process' switch.

```
LDA     2,DL
LDQ     =V18/FVECTR,18/IVECTR
MME1    64
```

FVECTR and IVECTR are locations in the '635 slave memory' at which the pseudo fault and interrupt vectors are located.

The Supplement changes the descriptor for segment 5 from slave procedure, slave access, write permit to master procedure.

The Supplement sets the '645 Pseudo-Process' switch so that if faults or interrupts (from the dedicated 645 pseudo-process GIOC) occur, the pseudo vectors provided by the user are interrogated (see later discussion on fault and interrupt handling).

c. Discontinue interrogation of pseudo fault vector.

```
LDA     3,DL
MME1    64
```

This request is honored only if the '645 Pseudo-Process' switch is on. The function provided is the ability to have faults interpreted by GECOS while executing "escape" coding (which really is 635 code).

d. Resume interrogation of pseudo fault vector.

```
LDA     4,DL
MME1    64
```

This request is honored only if the '645 Pseudo-Process' switch is on.

e. Store various hardware indicators.

```
LDA     5,DL
MME1    64
```

The Supplement reacts to this by storing the memory controller indicators (interrupt cells, access mask, and mask register) in the communication zone.

f. Location of Multics GIOC mailbox.

```
LDA     6,DL
LDQ     PTR,DU
MME1    64
```

Location PTR contains the absolute address of the mailbox area for 645 process work. See BE.7.13 for more information on direct utilization of a GIOC by a 645 pseudo-process.

When a fault occurs the Supplement checks a set of conditions and reacts accordingly. The specific checking is shown in the decision table diagrammed in Figure 2.

An example of the use of this diagram is: fault occurs, and execution mode at time of fault occurrence is append, and '645 Pseudo-Process' switch is off, then follow ACTION #3.

When an interrupt occurs on a device dedicated to GECOS, the Supplement reflects the interrupt to GECOS. When an interrupt occurs on the GIOC dedicated to 645 pseudo-process work, the Supplement checks a set of conditions and reacts and reacts accordingly. The specific checking is shown in the decision table diagrammed in Figure 3.

The '645 pseudo-process terminate' routine stores the Associative Memory, DBR Memory Controller Interrupt Cells, Memory Controller Access Masks, and Memory Controller Interrupt Mask Registers into the fixed Communication Region and transfers to the slave termination routine through a pointer in the Communication Region (see Figure 5).

(The Associative Memory and DBR are also stored in the Communication Region by the segment 'escape'.)

<table>
<tr><td rowspan="6">Occurrence of a fault</td><td rowspan="2">Execution Mode at time of fault occurrence is Absolute</td><td colspan="4">Fault Control-unit move switch is on. (Fault occurred while a prior fault was being simulated)</td><td>ACTION #1: Communicate trouble (code 102(8) to slave program, and go to 645 pseudo-process terminate routine</td></tr>
<tr><td colspan="4">Interrupt Control-unit move switch is on. (Fault occurred while an interrupt was being simulated)</td><td>ACTION #2: Communicate trouble (code 101(8) to slave program, and go to 645 pseudo-process terminate routine</td></tr>
<tr><td rowspan="4">Execution Mode at time of fault occurrence is Append</td><td colspan="4">Control-unit move switches are both off.</td><td rowspan="3">ACTION #3:
Reflect the fault to
GECOS</td></tr>
<tr><td colspan="4">'645 Pseudo-Process' switch is off.</td></tr>
<tr><td rowspan="3">'645 Pseudo-Process' Switch is on</td><td colspan="3">User's pseudo fault vector is inactive (MME1 64 type 3 in effect)</td></tr>
<tr><td rowspan="2">User's pseudo fault vector active</td><td>Instructions in user's pseudo fault vector are not SCU and TRA</td><td>timer runout</td></tr>
<tr><td>not timer runout</td><td>ACTION #4: Communicate fault information to slave program and go to the 645 pseudo-process terminate routine</td></tr>
<tr><td colspan="2">Instruction in user's pseudo fault vector are SCU and TRA</td><td>ACTION #5: Turn on the fault control-unit move switch. Move the control-unit; this simulates the users SCU (a fault can occur here resulting in ACTION #1). Turn off the fault control-unit move switch. Execute the user's TRA.</td></tr>
</table>

Figure 2. Fault handling in the GECOS Supplement - (Read from left to right)

<table>
<tr><td rowspan="4">Occurrence of an Interrupt</td><td colspan="3">Device on which interrupt occurred is dedicated to GECOS</td><td>ACTION #1:
Reflect the interrupt to GECOS</td></tr>
<tr><td rowspan="3">Device on which interrupt occurred is dedicated to 645 pseudo-process (GIOC)</td><td colspan="2">'645 Pseudo-Process' switch is off</td><td rowspan="2">ACTION #2:
Ignore the interrupt</td></tr>
<tr><td rowspan="2">'645 Pseudo-Process' switch is on</td><td>Instructions in users's pseudo interrupt vector are SCU and RCU</td></tr>
<tr><td>Instructions in user's pseudo interrupt vector are SCU and TRA</td><td>ACTION #3: Turn on the interrupt control-unit move switch. Move the control-unit; this simulates the users SCU (a fault can occur here resulting in fault ACTION #1). Turn off the interrupt control-unit move switch. Execute the user's TRA.</td></tr>
<tr><td></td><td></td><td>Instructions in user's pseudo interrupt vector are not SCU and TRA or SCU and RCU</td><td>ACTION #4: Communicate interrupt information to slave program and go 645 pseudo process terminate routine.</td></tr>
</table>

Figure 3. Interrupt Handling in the GECOS Supplement

(Read from left to right)

The termination routines, whether Supplement initiated or normal, have the responsibility for collecting information on machine conditions and writing this information, together with a core dump, onto a file (file code CR) in the same format used by the simulator.

Segment 'escape'

When entry point 'finish' is called, termination of the '645 Pseudo-Process' is effected by the normal termination routine. Segment 'escape' gets to this routine using escape number 0.

Escape to perform 635 escape coding is performed by saving the argument list pointer in segment 5 location 240, putting the escape number in index register 3, and transferring indirectly to segment 5 location 253.

Further details about the use of escape coding are available in MSPM BE.7.10.

2. The 645 Loader and Associated Routines

When the 645 Loader gets control, it has the responsibility of setting up a descriptor segment for the 645 Pseudo-Process, loading several fixed segments (described in Figure 4) and all other requested segments (details may be found in writeups on 6.36 and 64.5), setting up a communication region for the purpose of communicating with the Supplement and the 645 Pseudo-Process segment 'escape', and finally transferring control to the 645 Pseudo- Process in segment 'init'.

The fixed segments which are loaded are as follows:

Segment Number	Description	Descriptor Attributes
4	Pseudo fault vector at 400_8 in '635 slave memory'. Segment name is 'fvectr'. 192 words long, unpaged. First 64 words are the fault vector, remainder reserved for ITS pointers for SCU-TRA instruction pairs. Loader initializes to MME1 -1	Data
5	Describes the assigned 635 slave memory, unpaged, 1024 work blocks. Information obtained from Supplement (MME1 64). Segment name is 'memory'.	Slave procedure Slave access, write permit.
6	Describes the channel mailbox for the GIOC assigned to 645 Pseudo-Process work. Unpaged. Segment name 'gioc1'.	Data
7	Pseudo interrupt vector at 700_8 in '635 slave memory'. Segment name is 'ivectr'. 192 words long, unpaged. First 64 words are the interrupt vector(s), remainder reserved for ITS pointers for SCU-TRA instruction pairs. Loader initializes to SCU, RCU instruction pairs.	Data

Figure 4. Basic entries in the descriptor segment for a '645 Pseudo-Process'

Segment 'trace'

An instruction-by-instruction trace is available only under simulation. If the pseudo-op regarding an instruction-by-instruction trace (002(8)) were encountered under execution, it would cause the 645 pseudo-process to abort.

Segment 'trace' is available to issue the simulator's trace pseudo-op if the 645 pseudo-process is running under simulation, and to ignore trace requests if the pseudo-process is running under exection. This segment has just one entry point called 'trace' and it is called with one argument, which is zero to start tracing and nonzero to stop tracing (under simulation).

'trace' decides whether the pseudo-process is running under execution or simulation by testing location 'cacindxec' of segment 'cac635mem'.

The communication region which the 645 loader sets up has the following format:

Address within '635 slave memory'.		Length (words)	Description
decimal	octal		
192	300	32	Associative Memory
224	340	4	Unused
228	344	8	Memory Controller Interrupt Mask Registers and Access Mask Registers (format as by RMCM & STAQ) for four memory controllers.
236	354	4	Memory Controller Interrupt Cells for four Memory Controllers.
240	360	2	Area which segment 'escape' uses to pass a pointer (ITS pair) to the argument list.
242	362	2	ITS pair to return point in 645 pseudo-process. This is set by segment 'init-esc'. 635 escape coding returns to '645 Pseudo-Process' by executing an TRA 242,*instruction.
244	364	6	Area used by the Supplement to hold the control unit when the Supplement has determined that the '645 Pseudo-Process' should terminate itself.

Figure 5

Address within '635 slave memory'.		Length (words)	Description
decimal	octal		
250	372	1	Area used by the Supplement to indicate the fault number (format ARG N) or interrupt number (format ARG -N) causing the 645 process termination. N is the offset of the pair within the vector.
251	373	1	Indicator of '645 Pseudo-Process' run mode, execution versus simulation. (≠0 for execution.) Set by the 645 loader.
252	374	1	Unused
253	375	1	Pointer to 635 escape coding. (Set by 645 loader.)
254	376	1	Pointer to slave termination routine (i.e., an entrance to 636DMP) for the Supplement.
255	377	1	DBR contents at pseudo-process termination time.

Figure 5 (continued)