| bcc | title: Microprocessor Crash Procedures Initialization and Breakpoints | | prefix/class-number.revision: MPREC/W-30 | |
|---|---|---|---|---|
| checked: [illegible] | authors | | approval date: 9/30/69 | revision date |
| checked: [illegible] | Butler W. Lampson | | classification: Working Paper | |
| approved: Mel | [illegible] | | distribution: Company Private | pages: ~~8~~ 9 |

**ABSTRACT and CONTENTS**

Specifies the actions expected of all microprocessors in connection with system crashes and the earliest stages of initialization. Describes the uniform microcode which is used to handle the breakpoint feature.

## Introduction

The basic philosophy of the M1 system with regard to the detection and correction of system errors, both hardware and software, is that

1) As much checking as possible should be done for conditions which indicate the possibility of a system failure
2) When an error is detected, some standard action should be taken which should be reasonably flexible and (possibly complex) but which should be able to proceed even when most of the hardware is not working
3) This action will normally include an attempt to obtain and record as much information as possible about the failure
4) It will normally also include a complete cleanup of the system, so that execution can continue with some confidence that any errors in system tables have been corrected.

The ITP is the vehicle for most of the logging and recovery procedure, as described below.

## Hardware Failure Reporting

The system is equipped with five words worth of bits which are set as the result of detected malfunctions in the hardware; power failures, overheating, parity errors, etc. These words can be read by the UTP; their format is described in SR/S-8. When a word is read it is reset to Ø (although it may immediately become non-zero if error conditions persist). The union of all the error conditions except those in SWR5 is a signal called the system failure level or crash signal, which is used to set the UTP's STROBE2 flip-flop. Other microprocessors can also set this flip-flop by directing a STROBE2 to the UTP, and are expected to do so whenever they detect an error.

When the UTP sees its STROBE2, it enters a system recovery mode. Its first step in this mode is to send STROBE2 signals to all the other microprocessors. Each processor can test its STROBE2 with a branch condition, and is expected to do this as frequently as is convenient, but at least often enough to bring all activity to an orderly halt within 500 μs from the moment when the level appears. After bringing things to a halt in this way, the processor records any unusual conditions it knows about in an error reporting region (ERR) in core at a location peculiar to it. The processor then dumps its state as for a break and hangs until the break wait location (BRKWAIT), which is another core location peculiar to it, becomes non-zero. It zeros this location before hanging. When BRKWAIT = 12343210B, it reloads the state (exactly as though a break had completed) and starts executing at the location given by the break address word BRKADR.

The CPU follows a slightly different procedure; it treats crash exactly like single-step.

## Recovery Action

The UTP plays a special role in the handling of failures. When it sees the crash signal, instead of going into a wait state like the other processors, it turns control over to an ITP program which tries to figure out what is going on. This is done in a rather cautious fashion, however, to avoid getting into trouble if critical parts of the system are not working.

1) First the system warning registers are checked to make sure that power to the UTP and core is not about to fail. If it is, bits Ø-11 in SSL are turned on and the processor hangs.

2) A special microcode sequence is entered which tests the memory for reliable operation. If the memory proves to be so faulty that an ITP program cannot execute, it turns on bits 12-23 in SSL and hangs. If it appears the the memory could function in 4-module mode, some as yet undefined action is taken. If the memory is OK,

3) Schedule mode is turned off and the ITP reset sequence is executed. This will start the ITP running, and it can proceed to determine the magnitude of the problem and decide what to do about it.

## System Restart

As a result of a crash signal, or perhaps for other reasons, it may be desirable for the ITP to restart one or all of the processors. To this end it has the ability to send a ZM signal (clear 0) to any of them. In this way the system restart procedure described elsewhere can readily be initiated.

To send ZM to other processors, the ITP should address device UPZM with a POT instruction and transmit a data word which has the same form as the word which a microprocessor puts on the X bus when it sends a STROBE.

## Initialization and Breakpoints

When a microprocessor (other than the CPU) finds itself at location Ø, it proceeds as follows:

1) Check for a break (branch condition BREAK=1, #45). If there is a break, store the state at SAVE in the following format

```
M
SK1 - SKn       (n = 31 or 63)
RØ - R6
OS
Q
Z
```

then wait for BRKWAIT to become non-zero. When it does, reload the state, take the new value for the O register from BRKADR and resume execution.

Code to handle this aspect of initialization is listed below. It may be found on (LAMPSON)BREAK. It saves M in SKØ and exchanges the scratchpad with core to save or restore it. Sending control to LOADST loads the state.

2) If there is no break, wait for a STROBE.
3) When the STROBE is received, take any special initialization action which may be required.
4) Then load state as if returning from a break.

## Fixed Core Addresses and Parameters

| | |
|---|---|
| Microprocessor numbers N: | 1 = AMC |
| | 2 = UTP |
| | 3 = CHIO |
| | 4 = CPUØ |
| | 5 = CPU1 |

```
BRKADR    =  20B + (N-1)    (not for CPUs)
BRKWAIT   =  23B + (N-1)
ERR       =  244ØB + (N-1)*4
SAVE      =  25ØØ + (N-1)*12ØB    (not for CPUs)
```

```
PAGE   1     SYSTEM     (HECKEL)/BREAK     09/26/69 0647!44

* PARAMETERS DEFINING THE SAVE AREA, USED BY THE STATE STORE AND
* LOAD ROUTINES
       DP SAVE√2620B;            * FOR UTP, SEE SYSP/W-15
       DP SAVER0√SAVE+LSCRATCH;
* THE CODE SAVES R0-R6, ΘS, QAND Z IN SEQUENCE
       DP SAVER1√SAVER0+1;
       DP SAVEZ√SAVER0+9;
* A SCRATCHPAD LOCATION MUST BE DEDICATED TO SAVING M
       DE SK0 AS SSMREG;
* THE Θ REGISTER IS RESTORED FROM
       DP BREAK√21B;              * FOR UTP, SEE SYSP/W-15
* AND WE WAIT AFTER STORING THE STATE FOR
       DP BRKWAIT√26B;            * FOR UTP, SEE SYSP/W-15
* TO BECOME NON-ZERO BEFORE STARTING TO RELOAD

* BREAK. DUMP STATE IN THE SAVE AREA.  THE STRATEGY IS AS FOLLOWS
* 1) SAVE M IN SSMREG IN THE SCRATCHPAD
* 2) STORE MAR AT SAVER0. STORE ΘS AT SAVEΘS
* 3) STORE Z,Q AT SAVEZ,Q AND R1-R6 AT SAVER1 TO SAVER6
* 4) EXCHANGE SCRATCHPAD AND SAVE TO SAVE+LSCRATCH-1

* AFTER SAVING THE STATE, BREAK WAITS UNTILL BRKWAIT
* BECOMES NON-ZERO.

* THE RELOAD STRATEGY IS OBTAINED BY DOING STEPS 4-3 IN REVERSE,
* AND THEN
* 2) FETCH SAVEΘS AND DGOTO IT
* 1) DGOTO *+1 (THIS SETS UP ΘS), FETCH BREAK
* 0) DGOTO M (THIS SETS UP Θ), FETCH SAVER0
* -1) MAR√M, M√SSMREG

       MACRO IMS√MAR√Z√Z+1, STORE;
SAVEST! MAR√SAVER0, STORE;
       M√Z, MAR√SAVEZ, STORE;
       M√R1, MAR√Z√SAVER1, STORE;
       M√R2, IMS;
       M√R3, IMS;
       M√R4, IMS;
       M√R5, IMS;
       M√R6, IMS;
       M√ΘS, IMS;
       M√Q, MAR√Z√MAR+1, STORE;
       MAR√SAVE-1;
       R2√-LSCRATCH, Z√0, CALL XSCRATCH;
BWAIT! MAR√BRKWAIT, FETCH;
       GOTO *-1 ON M=0, R2√-LSCRATCH;
LOADST! MAR√SAVE-1, Z√0, CALL XSCRATCH;
       MAR√SAVER1, FETCH;
       R1√M, CALL FN;
       R2√M, CALL FN;
       R3√M, CALL FN;
       R4√M, CALL FN;
       R5√M, CALL FN;
       R6√M, CALL FN;
```

```
PAGE   2   SYSTEM    (HECKEL)/BREAK    09/26/69 0647:44

        DGØTØ M, MAR√MAR+1, FETCH;
        Q√M, MAR√MAR+1, FETCH, CALL *+1;
        Z√M, MAR√BREAK, FETCH;
        DGØTØ M, MAR√SAVERO, FETCH;
        MAR√M, M√SSMREG;

* SUBRØUTINE TØ BUMP MAR AND FETCH
FN:     MAR√MAR+1, FETCH, RETURN;

* SUBRØUTINE TØ EXCHANGE -(R2) SCRATCHPAD LØCATIØNS, STARTING AT (Z),
* WITH CØRE LØCATIØNS STARTING AT (MAR)+1. CLØBBERS M,Z,R1,R2
XSCRATCH: FETCH, MAR√MAR+1;
        R1√SKZ;
        SKZ√M, M√R1, DGØTØ XSCRATCH;
        STØRE, Z√Z+1, RETURN ØN R2√R2+1>=0;
```