TO: MSPM Distribution
FROM: D. R. Widring
SUBJ: BL.8.00
DATE: 08/17/67

Minor changes have been made to reflect current implementation. Also, the initialization of Hard-core static storage is treated as a separate step.

Published: 08/17/67
(Supersedes: BL.8.00, 06/07/67)

Identification

Hardcore I/O Initialization
R. C. Daley, D. R. Widrig

Purpose

This section provides the specification of the procedures responsible for the initialization of the hardcore I/O system. These procedures are run under the control of the Multics Initializer control program. In addition, a special initialization procedure for use in assigning GIOC channels during the early stages of Multics initialization is specified.

Introduction

The initialization of the hardcore I/O system is accomplished in two stages. During the first stage, the GIOC interface module (GIM) and its data bases are initialized for subsequent use by the file system. During this stage, an I/O device configuration table (DCT) is constructed to contain entries for only those GIOC channels to be used during system initialization. Since, at this stage of initialization, the file system dynamic paging mechanism has not yet been initialized, the entire DCT, which may be quite large, cannot be constructed at this time.

The second stage of hardcore I/O initialization is called upon after the file system has been initialized and the dynamic paging mechanism has been established. At this time the construction of the DCT is completed.

Many of the data bases of the GIM are initialized as needed during the normal operation of these modules. As a result, no explicit action to initialize these data bases is required.

Initialization Procedures

At the appropriate point during Multics initialization, the Multics Initializer control program begins the initialization of the hardcore I/O system by means of the following call.

```
call io_init$one;
```

When this call is received, the followin steps are taken to initialize the hardcore I/O system.

Step 1

Static storage for the entire Hard-core I/O system is initialized by means of the following call:

```
call init_hcio_stat;
```

Further references to the function of the Hard-core I/O system static storage may be found in BF.20.09.

Step 2

The Channel Assignment Table (CAT) is partially initialized by means of the following call;

```
call init_cat$one;
```

The CAT (See BF.20.09 for a detailed description of the data base) is divided into two major sections; the per-GIOC section and the per-device section. Initialization of the per-device section requires only that all per-device entries be zero. Since the CAT is created as an all-zero segment, the per-device section is, by default, in an initialized state. Only the per-GIOC section associated with the bootload GIOC is initialized during the init_cat$one call.

This procedure initializes the GIOC mail box pointers and GIOC adapter tables in the CAT from information contained in the system configuration tables.

The various channels of each GIOC available to Multics are initialized by placing appropriate command words in their mail boxes. No connects are issued. Rather the commands are left to be discovered by the separate channels. The commands are as follows:

1. The connect channels are initialized by placing a zero connect pointer word (CPW) in each connect mailbox. This action will nullify any outstanding connect sequences. At this time the connect tables in the CAT are initialized. Refer to BF.20.11 for a discussion of connect tables.

2. The data channels are initialized by placing a command DCW with bit 17 ON in each data mailbox. This command DCW serves to initialize the status channel pointers for the data channel and stops any active channel on its next request for service to the GIOC.

3. The status channels are initialized by inserting proper addresses and tallies in the status control words in each status channel mailbox. At this time the status channel tables in the CAT are initialized. Refer to BF.20.12 for a discussion of status channel tables.

Step 3

A basic I/O device configuration table (DCT) is constructed by means of the following call.

```
call init_dct$part_1;
```

This procedure constructs a basic DCT from symbolic information contained in the ASCII segment "dct_sym_1". The basic DCT must contain entries for all GIOC channels used by the basic file system and at least one GIOC channel through which the Multics system tape may be accessed.

Control is returned to the Multics Initializer control program. The GIM is now available for use. However, any GIOC channel to be used between the first and second stages of hardcore I/O initialization must be initialized by means of the special initialization procedure described below, rather than through the normal GIM calls. This restriction is imposed since the normal GIM calls for channel initialization require the use of the file system which is not yet initialized.

Stage 2

After the file system has been initialized and the dynamic paging mechanism has been established, the Multics Initializer control program continues hardcore I/O initialization by means of the following call.

```
call io_init$two;
```

When this call is received, initialization of the hardcore I/O system continues with the following steps.

Step 1

During the second half of Hard-core I/O system initialization, the remainder of the per-GIOC data entries in the CAT are initialized via the following call:

```
call init_cat$two;
```

Recalling that the earlier call to init_cat$one initialized all relevant data bases for the bootload GIOC, it is assumed that possible operator interactions, etc, have completely specified all GIOC configurations to be operated by the Hard-core I/O system. These new entries are initialized now.

Step 2

The remainder of the DCT is constructed by means of the following call.

```
call init_dct$part2;
```

This procedure constructs the remainder of the DCT from symbolic information contained in the ASCII segment "dct_sym_2". Hardcore I/O initialization is now complete.

Special Initialization Procedure

During normal Multics operation a GIOC channel is initialized by the GIM via a call to define$channel followed by a call to define$class. When these calls are received by the GIM, the GIM calls upon the basic file system to create a logical channel table (LCT) and find the class driving table (CDT) to be used in the operation of the channel (see section BF.20.05). However, certain initialization procedures and, in fact, the file system itself must be able to make use of the GIM before the file system is fully initialized. For this reason, the following initialization procedure is provided for use only during system initialization.

```
call define$assign (chan_name,dev_index,lctno,
                    dctno,rtnstat);
```

This call provides the same effect as a call to define$class. The arguments for this call are declared in the following PL/I statement and are described in detail below.

```
dcl chan_name char (*),

    dev_index fixed bin (17),

    lctno bit (18),

    cdtno bit (18),

    rtnstat bit (36),
```

chan name is a string of up to 32 characters specifying the desired GIOC channel.

dev index is the device index (see BD.8.03) returned to the caller and defines the event cell to be set when a hardware interrupt occurs.

lctno specifies the segment number of an empty segment to be used as the LCT for this channel.

cdtno specifies the segment number of the CDT to be used in constructing DCW's for this channel.

rtnstat is the standard status bit string returned on normal calls to the GIM (see section BF.20.05).