HONEYWELL INTEROFFICE CORRESPONDENCE

PHOENIX OPERATIONS – HONEYWELL INFORMATION SYSTEMS

DATE 771108 PHONE MAIL ZONE COPIES

TO Specification File

FROM Conrad Bjerke

COMPONENT LADC

SUBJECT CP-6 I/O System

The attached changes for doc. # 0160B, reflect the current implementation of IOQ.

Please replace the following pages with the attached updates:

4-10
12-17

Please remove pages 11 and 13a.

Conrad C. Bjerke

Conrad Bjerke

HONEYWELL INTEROFFICE CORRESPONDENCE

PHOENIX OPERATIONS – HONEYWELL INFORMATION SYSTEMS

DATE 770620 PHONE MAIL ZONE COPIES

TO Architecture Fi[illegible]

FROM Conrad Bjerke/John Catozzi

COMPONENT LADC

SUBJECT CP-6 I/O System (IOQ)

SF section: 14.4
doc # : 0160B-0
page : 1

1. INTRODUCTION

The CP-6 I/O System (IOQ) will provide all input-output facilities for disk storage, magnetic tape, local card readers and punches, local line printers, and the system console. In addition, it will provide connect and interrupt facilities for the direct channel to the L6 Front End.

This document describes the release version of IOQ. For initial testing there will be an interim version based on GCOS IOS. This interim version will have the Monitor interfaces described in Sections 2, 3, and 4, but it will not have all of the error recovery or Test and Diagnostics (T&D) facilities described in Sections 5 and 6. As IOQ evolves from the interim version to the release version, these facilities will be added.

Sections 2, 3, and 4 describe the interface which will be used for all normal I/O, i.e., all I/O except T&D. Section 2 is a functional overview, Section 3 describes the I/O request packet, and Section 4 describes the calling sequences.

Section 5 describes the error recovery and logging techniques to be used by IOQ.

Section 6 describes the T&D interface to IOQ.

2. FUNCTIONAL OVERVIEW OF "NORMAL-MODE" SERVICES

Functionally the services provided by CP-6 IOQ are similar in nature to the CP-V IOQ services. Callers specify logical function codes and device indices along with buffer description and end-action information. IOQ then calls a device pre-handler to build the command list needed to perform the operations, starts the I/O, and upon completion reports the type completion code to the caller via the user's DCB or end-action routine. The major function differences arise from the way information is passed to and returned from IOQ.

In CP-6, the IOQ tables consist of a pool of queue entry packets. To issue an I/O request one first requests a free queue entry or packet. The request routine returns a PL-6 pointer to the packet, which the caller uses with a based structure template to stuff the packet. The pointer to the packet is then passed to the queuing routine which processes the request. After post-processing of the I/O operation completion, if end-action was requested by the caller, the packet (now including completion status like type completion and actual record size) is passed to the end-action routine. Upon return from the end-action, the packet is either freed or re-queued, depending on a flag set by the end-action routine.

The second source of difference arises from a generalization of the AVR service in CP-V. On the L66 system all devices generate special interrupts - interrupts not necessarily the result of an I/O operation - such as when a device changes state from manual to ready. In CP-6, IOQ will provide a generalized service allowing monitor routines to gain control of these interrupts on a selective basis by device index.

IOQ provides for both mapped and non-mapped I/O, either user-associated or not, wait or no-wait service, DCB posting, event posting, end-action control, and special interrupt control. The following sections describe each of these services.

2.1 Non-User Associated I/O

This service is exactly like CP-V's NEWQNW. No system scheduler service is involved, the I/O is always queued and an immediate return made to the caller. At completion, the specified end-action routine is called. Using a buffer pointer format of BUFPPNO, any page in the first 16M words of real memory may be addressed. It is also possible to perform non-user associated I/O, using a users map (such as for "swapping"). The use of a map for the I/O is dependent only

on the descriptor of the buffer as referenced by the PL-6 pointer. Non-user associated requests imply no system scheduler interfaces and no posting of status in a DCB.

2.2 User-Associated I/O

This service provides an interface which is a logical combination of NEWQ and QUEUE. User-associated I/O implies system scheduler interfaces such as Master Function Count check and the possibility of services like waiting on the I/O operation, posting status in the user's DCB and reporting event completion. Master Function Count check is always performed for user-associated I/O resulting in blocking the user if he has exceeded the master function limit for his job class (online or batch). The other functions of this service request are selectively enabled by bit flags in the request packet or by the parameters of the call. (See Section 4. - Calling Sequences - for details.)

2.3 Special Interrupt Control

The device tables will contain the address to which control will be transferred upon receipt of special interrupts. This address may be separately established for each device by a call on IOQ specifying the device index (DCTX) and the entry point to which control is to be passed. At the time of a special interrupt, if such control has been established, the DCTX and the interrupt status word will be passed to the specified address for processing. The special interrupt request will remain in effect until an explicit call is made to IOQ to disable the request.

It is conceivable that a monitor service could be provided to allow a user to get control (à la M:INT) for special interrupts on a device currently allocated to him.

2.4 Cleanup of User-associated I/O

There will be an entry provided to clean up all I/O for a user job when that job terminates.

3. QUEUE PACKET FORMAT

```
DCL 1 NI$REQ BASED ALIGNED,                    /* I/O REQUEST PACKET */
    0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8
   ┌─────────────────┬─────────────────┬─────────────────┬─────────────────┐
 0 │FL$
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
 1 │DLA
   │SETX             │SETADDR
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
 2 │BUFSIZE                                │*│FC       │OPFLG
   │                                       │ │         │U│A│E│E│W│B│H│R│S│
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
 3 │BUF$
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
 4 │PTP                                    │*
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
 5 │DCB$
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
 6 │EAENTRY
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
 7 │EAINFO
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
10 │EVNTINFO
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
11 │ARSIZE                                 │*│CC       │USER#
   │                                       │ │ERR  │ABN  │
   │                                       │ │I│P│I│E│E│B│
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
12 │ARCT
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
13 │RCT
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
14 │LPWX
   │BASE                               │SIZE
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
15 │SCT                                │ADCW2
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
16 │DCWLIST
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
   LEVEL 2 ARRAY:   6 ('   6'O) ENTRIES TOTAL.
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
24 │SEEK
   │SCL                    │SSZ    │SECTOR
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
25 │CL$
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
26 │STATUS1
   │P│P│MAJOR  │MINOR      │*          │IOM        │*          │RCR
   ├─────────────────┬─────────────────┬─────────────────┬─────────────────┤
27 │STATUS2
   │NXTADDR                            │NXTCH│R│*
   └─────────────────┴─────────────────┴─────────────────┴─────────────────┘
```

```
2 FL$ PTR,                          /* LINK TO NEXT PACKET IN QUEUE */
2 DLA ALIGNED,                      /* DEVICE LOGICAL ADDRESS */
  3 SETX UBIN(9) UNAL,              /* DEVICE OR SET INDEX */
  3 SETADDR UBIN(27) UNAL,          /* DEVICE- OR SET-RELATIVE ADDRESS */
2 BUFSIZE UBIN(20) UNAL,            /* BUFFER SIZE (BYTES) */
2 * BIT(1) UNAL,
2 FC UBIN(6) UNAL,                  /* LOGICAL FUNCTION CODE */
2 OPFLG,                            /* OPERATION FLAGS */
  3 USER BIT(1) UNAL,               /* USER-ASSOCIATED I/O */
  3 ARS BIT(1) UNAL,                /* SET ARS OF DCB */
  3 EVNT BIT(1) UNAL,               /* REPORT COMPLETION EVENT */
  3 EA BIT(1) UNAL,                 /* CALL END-ACTION ROUTINE */
  3 WAIT BIT(1) UNAL,    /* BLOCK ASSOCIATED USER AFTER QUEUEING REQUEST
  3 BPF BIT(1) UNAL,                /* BUF$ CONTAINS PTR -> BUFFER */
  3 HOLD BIT(1) UNAL,               /* DO NOT RELEASE PACKET ON COMPLETION
  3 REQ BIT(1) UNAL,                /* RE-QUEUE PACKET */
  3 SET BIT(1) UNAL,                /* 0=REQ TO DEV, 1=REQ TO SET */
2 BUF$ PTR,                         /* "BUF$ PTR" IF BPF='1'B */
                                    /* "BUFADDR UBIN(26)" IF BPF='0'B */
2 PTP UBIN(18) UNAL,                /* BPF=0: PAGE TABLE POINTER (0=REAL) *
2 * SBIN HALF UNAL,
2 DCB$ PTR,                         /* -> USER'S DCB */
2 EAENTRY EPTR,                     /* END-ACTION PROCEDURE */
2 EAINFO UBIN(36),                  /* END-ACTION PARAMETER */
2 EVNTINFO UBIN(36),                /* EVENT INFO */
2 ARSIZE UBIN(20) UNAL,             /* ACTUAL RECORD SIZE */
2 * BIT(1) UNAL,
2 CC,                               /* LOGICAL COMPLETION CODE (0=OK) */
  3 ERR,
    4 IO BIT(1) UNAL,               /* I/O ERROR    */
    4 PE BIT(1) UNAL,               /* PARITY ERROR */
    4 INVD BIT(1) UNAL,             /* INVALID OPERATION */
  3 ABN,
    4 EOF BIT(1) UNAL,              /* END-OF-FILE */
    4 EOT BIT(1) UNAL,              /* END-OF-TAPE */
    4 BOT BIT(1) UNAL,              /* BEGINNING-OF-TAPE */
2 USER# UBIN(9) UNAL,               /* USER ID */
2 ARCT SBIN UNAL,                   /* ACTUAL RECORD COUNT */
2 RCT SBIN UNAL,                    /* RECORD COUNT */
2 LPWX ALIGNED,                     /* LIST POINTER WORD EXTENSION */
  3 BASE UBIN(18) UNAL,             /* LOWER BOUND (MOD 2 WORDS) */
  3 SIZE UBIN(18) UNAL,             /* SIZE (WORDS) */
2 SCT UBIN(18) UNAL,                /* DEVICE TABLE ADDRESS */
2 ADCW2 UBIN(18) UNAL,              /* ADDRESS OF SECOND DCW LIST */
2 DCWLIST(0:5) UBIN(36),            /* DCW LIST */
2 SEEK ALIGNED,                     /* DISK SEEK ADDRESS WORD */
  3 SCL UBIN(12) UNAL,              /* SECTOR COUNT LIMIT */
  3 SSZ UBIN(4) UNAL,               /* SECTOR SIZE */
  3 SECTOR UBIN(20) UNAL,           /* SEEK ADDRESS */
2 CL$ PTR,                          /* LINK TO DCW LIST PACKETS */
2 STATUS1 ALIGNED,                  /* STATUS WORDS */
  3 PRESENCE BIT(1) UNAL,
```

```
   3 POWEROFF BIT(1) UNAL,          /* POWER OFF STATUS */
   3 MAJOR BIT(4) UNAL,             /* DEVICE MAJOR STATUS */
   3 MINOR BIT(6) UNAL,             /* DEVICE MINOR STATUS */
   3 * BIT(6) UNAL,
   3 IOM BIT(6) UNAL,               /* IOM / CHANNEL STATUS */
   3 * BIT(6) UNAL,
   3 RCR UBIN(6) UNAL,              /* RECORD COUNT RESIDUE */
 2 STATUS2 ALIGNED,                 /* 2ND STATUS WORD */
   3 NXTADDR UBIN(18) UNAL,         /* NEXT ABS ADDRESS */
   3 NXTCHAR UBIN(3) UNAL,          /* NEXT CHAR POSITION */
   3 READFLG BIT(1) UNAL,           /* READ/WRITE FLAG */
   3 * BIT(14) UNAL;
```

```
DCL 1 NI$VREQ BASED ALIGNED,                    /* I/O REQUEST PACKET */
    0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8
   +-----------------+-----------------+-----------------+-----------------+
 0 |FL$
   +-----------------+-----------------+-----------------+-----------------+
 1 |DLA
   |DCTX               |DRELADDR
   +-----------------+-----------------+-----------------+-----------------+
 2 |BUFSIZE                                  |*|FC           |OPFLG
   |                                         | |             |U|A|E|E|W|B|H|R|S|
   +-----------------+-----------------+-----------------+-----------------+
 3 |BUFADDR                                                  |# # # # # # # # #
   +-----------------+-----------------+-----------------+-----------------+
 4 |PTP                                      |*
   +-----------------+-----------------+-----------------+-----------------+
 5 |DCB$
   +-----------------+-----------------+-----------------+-----------------+
 6 |EAENTRY
   +-----------------+-----------------+-----------------+-----------------+
 7 |EAINFO
   +-----------------+-----------------+-----------------+-----------------+
10 |EVNTINFO
   +-----------------+-----------------+-----------------+-----------------+
11 |ARSIZE                                   |*|CC           |USER#
   |                                         | |ERR  |ABN    |
   |                                         | |I|P|I|E|E|B|
   +-----------------+-----------------+-----------------+-----------------+
12 |ARCT
   +-----------------+-----------------+-----------------+-----------------+
13 |RCT
   +-----------------+-----------------+-----------------+-----------------+
14 |LPWX
   |BASE                                     |SIZE
   +-----------------+-----------------+-----------------+-----------------+
15 |SCT                                      |ADCW2
   +-----------------+-----------------+-----------------+-----------------+
16 |DCWLIST
   +-----------------+-----------------+-----------------+-----------------+
   LEVEL 2 ARRAY:   6 ('   6'O) ENTRIES TOTAL.
   +-----------------+-----------------+-----------------+-----------------+
24 |SEEK
   |SCL                    |SSZ    |SECTOR
   +-----------------+-----------------+-----------------+-----------------+
25 |CL$
   +-----------------+-----------------+-----------------+-----------------+
26 |STATUS1
   |P|P|MAJOR  |MINOR      |*          |IOM        |*      |RCR
   +-----------------+-----------------+-----------------+-----------------+
27 |STATUS2
   |NXTADDR                                  |NXTCH|R|*
   +-----------------+-----------------+-----------------+-----------------+
```

```
2 FL$ PTR,                          /* LINK TO NEXT PACKET IN QUEUE */
2 DLA ALIGNED,                      /* DEVICE LOGICAL ADDRESS */
  3 DCTX UBIN(9) UNAL,              /* DEVICE OR SET INDEX */
  3 DRELADDR UBIN(27) UNAL,         /* DEVICE- OR SET-RELATIVE ADDRESS */
2 BUFSIZE UBIN(20) UNAL,            /* BUFFER SIZE (BYTES) */
2 * BIT(1) UNAL,
2 FC UBIN(6) UNAL,                  /* LOGICAL FUNCTION CODE */
2 OPFLG,                            /* OPERATION FLAGS */
  3 USER BIT(1) UNAL,               /* USER-ASSOCIATED I/O */
  3 ARS BIT(1) UNAL,                /* SET ARS OF DCB */
  3 EVNT BIT(1) UNAL,               /* REPORT COMPLETION EVENT */
  3 EA BIT(1) UNAL,                 /* CALL END-ACTION ROUTINE */
  3 WAIT BIT(1) UNAL,    /* BLOCK ASSOCIATED USER AFTER QUEUEING REQUEST *
  3 BPF BIT(1) UNAL,                /* BUF$ CONTAINS PTR -> BUFFER */
  3 HOLD BIT(1) UNAL,               /* DO NOT RELEASE PACKET ON COMPLETION *
  3 REQ BIT(1) UNAL,                /* RE-QUEUE PACKET */
  3 SET BIT(1) UNAL,                /* 0=REQ TO DEV, 1=REQ TO SET */
2 BUFADDR UBIN(26),                 /* "BUF$ PTR" IF BPF='1'B */
                                    /* "BUFADDR UBIN(26)" IF BPF='0'B */
2 PTP UBIN(18) UNAL,                /* BPF=0: PAGE TABLE POINTER (0=REAL) */
2 * SBIN HALF UNAL,
2 DCB$ PTR,                         /* -> USER'S DCB */
2 EAENTRY EPTR,                     /* END-ACTION PROCEDURE */
2 EAINFO UBIN(36),                  /* END-ACTION PARAMETER */
2 EVNTINFO UBIN(36),                /* EVENT INFO */
2 ARSIZE UBIN(20) UNAL,             /* ACTUAL RECORD SIZE */
2 * BIT(1) UNAL,
2 CC,                               /* LOGICAL COMPLETION CODE (0=OK) */
  3 ERR,
    4 IO BIT(1) UNAL,               /* I/O ERROR    */
    4 PE BIT(1) UNAL,               /* PARITY ERROR */
    4 INVD BIT(1) UNAL,             /* INVALID OPERATION */
  3 ABN,
    4 EOF BIT(1) UNAL,              /* END-OF-FILE */
    4 EOT BIT(1) UNAL,              /* END-OF-TAPE */
    4 BOT BIT(1) UNAL,              /* BEGINNING-OF-TAPE */
2 USER# UBIN(9) UNAL,               /* USER ID */
2 ARCT SBIN UNAL,                   /* ACTUAL RECORD COUNT */
2 RCT SBIN UNAL,                    /* RECORD COUNT */
2 LPWX ALIGNED,                     /* LIST POINTER WORD EXTENSION */
  3 BASE UBIN(18) UNAL,             /* LOWER BOUND (MOD 2 WORDS) */
  3 SIZE UBIN(18) UNAL,             /* SIZE (WORDS) */
2 SCT UBIN(18) UNAL,                /* DEVICE TABLE ADDRESS */
2 ADCW2 UBIN(18) UNAL,              /* ADDRESS OF SECOND DCW LIST */
2 DCWLIST(0:5) UBIN(36),            /* DCW LIST */
2 SEEK ALIGNED,                     /* DISK SEEK ADDRESS WORD */
  3 SCL UBIN(12) UNAL,              /* SECTOR COUNT LIMIT */
  3 SSZ UBIN(4) UNAL,               /* SECTOR SIZE */
  3 SECTOR UBIN(20) UNAL,           /* SEEK ADDRESS */
2 CL$ PTR,                          /* LINK TO DCW LIST PACKETS */
2 STATUS1 ALIGNED,                  /* STATUS WORDS */
  3 PRESENCE BIT(1) UNAL,
```

```
   3 POWEROFF BIT(1) UNAL,          /* POWER OFF STATUS */
   3 MAJOR BIT(4) UNAL,             /* DEVICE MAJOR STATUS */
   3 MINOR BIT(6) UNAL,             /* DEVICE MINOR STATUS */
   3 * BIT(6) UNAL,
   3 IOM BIT(6) UNAL,               /* IOM / CHANNEL STATUS */
   3 * BIT(6) UNAL,
   3 RCR UBIN(6) UNAL,              /* RECORD COUNT RESIDUE */
 2 STATUS2 ALIGNED,                 /* 2ND STATUS WORD */
   3 NXTADDR UBIN(18) UNAL,         /* NEXT ABS ADDRESS */
   3 NXTCHAR UBIN(3) UNAL,          /* NEXT CHAR POSITION */
   3 READFLG BIT(1) UNAL,           /* READ/WRITE FLAG */
   3 * BIT(14) UNAL;
```

3.1 Logical Function Codes

NAME	DEFINITION	CR	CP	LP	DP	MT	SC
NI_RQSTAT	Request status	X	X	X	X	X	X
NI_RDBIN	Read binary	X			X	X	
NI_RDASC	Read ASCII	X				X	X
NI_WRBIN	Write binary		X		X	X	
NI_WRASC	Write ASCII		X	X		X	X
NI_WRCMP	Write and compare				X		
NI_WRASCED	Write ASCII edited			X			
NI_ALARM	Turn on operator alarm						X
NI_SLEW	Slew			X			
NI_LDVFC	Load VFC image			X			
NI_LDCHAIN	Load print chain image			X			
NI_RDSTREG	Read status register			X	X	X	
NI_RDEBC	Read EBCDIC					X	
NI_WREBC	Write EBCDIC					X	
NI_WRFM	Write file mark					X	
NI_SPREC	Space records					X	
NI_SPFILE	Space files					X	
NI_SD800	Set density=800					X	
NI_SD1600	Set density=1600					X	
NI_SD6250	Set density=6250					X	
NI_REWIND	Rewind					X	
NI_UNLOAD	Rewind and unload					X	
NI_PERMIT	Tape write permit					X	
NI_PROTECT	Tape write protect					X	

The status codes with "TYC" in the type column have a one-to-one correspondence with bits in the TYC field of DCB and will be reported therein.

4. CALLING SEQUENCES

4.1 NIQ$GET

Call form:

```
CALL NIQ$GET(Q$);
```

Purpose:

NIQ$GET obtains a pointer to a queue entry packet. The queue format is described in Section 3. A based structure is used by the requesting routine to fill in the appropriate information.

If no packets are available and there is a current user, a "no packet available" event will be reported to the scheduler and the user will be blocked. If there is no current user, NIQ$GET will loop, trying to start I/O, until a packet becomes available.

4.2 NIQ$REL

Call form:

```
CALL NIQ$REL(Q$);
```

Purpose:

NIQ$REL frees the packet specified by Q$. This routine would normally be used only by IOQ, but others could use it.

If this packet is now the only available packet the scheduler is called to report "queue packet available".

4.3 NIO$DEVIO

Call form:

```
CALL NIO$QUE(Q$)ALTRET(label);
```

where Q$ is the PTR returned from a call on NIQ$GET.

Purpose:

NIO$DEVIO queues a request for I/O on a particular logical device index (DCTX).

To issue an I/O request, one calls NIQ$GET which returns a pointer to the queue packet to be used. Various fields of the queue packet (as described by a based structure) must be filled in specifying the operation to be performed. (See Section 3.)

The alternate return is taken if:

- o DCTX is not a valid device table index.
- o DRELADDR is not a valid address on this disk.
- o FC is not a valid logical function code.

In addition to the above reasons, the alternate return is taken if OPFLG.SET = '1'B and:

- o SETX is not a valid pack set index.
- o SRELADDR is not a valid address in this packet set.

4.4 End-Action Routine

Call form:

```
CALL EAENTRY(Q$);
```

where:

EAENTRY is the end action routine specified in the queue packet.

Q$ is a PL-6 pointer to the queue packet.

When the end-action routine is called, the buffer address will be in BUFADDR and PTP of the queue packet and OPFLG.BPF will be set to '1'B.

OPFLG.REQ in the queue packet will be set to '0'B by IOQ. The end action routine may set this flag to '1'B to indicate to IOQ that the packet is to be requeued. If OPFLG.REQ is set to '0'B, the disposition of the packet is controlled by OPFLG.HOLD. If the latter is '1'B, the packet will not be released. In this case, it is up to the caller to release the packet when he is done with it.

4.5 Scheduler Interfaces

IOQ interfaces with the scheduler to provide three basic functions. These are:

- Ability to wait on user associated I/O. This means that the call on NIO$DEVIO or NIO$SETIO does not return until the I/O operation is completed.

- Ability to limit the number of outstanding I/O requests for a user. This check - called Master Function Check - is always invoked for user associated I/O and causes the user to be blocked until previous requests complete. The I/O request which causes the block is not queued until after the scheduler determines that the Master Function Count is below the users threshold.

- Ability to cause asynchronous reporting of an I/O complete event to the user's program.

- Report that no queue packets are available. The current user is to be blocked until a queue packet becomes available.

- Report that a queue packet is now available. This allows the scheduler to unblock all users who were previously blocked for lack of a queue packet.

The calls to the scheduler to provide these functions are:

```
CALL SSR$REG(SS_IIP,DCB$);
```

Place the current user in the I/O in progress state until I/O complete occurs.

SS_IIP is the I/O In Progress event code.

DCB$ is the PL-6 PTR to the user's DCB.

```
CALL SSR$REG(SS_QMF);
```

Block the current user because his Master Function Count is exceeded.

SS_QMF is the Queue for Master Function event code.

```
CALL SSR$RUE([SS_IOC,SS_IOCU],USR#,DCB$,EVNTIFO);
```

Report I/O complete event to the scheduler.

SS_IOC is the I/O Complete event code.

SS_IOCU is the I/O Complete and unblock event code.

USR# is the user# of the user associated with the I/O.

DCB$ is the PL-6 pointer to the user's DCB.

EVNTINFO is 72 bits of information including TYC and the user's EVNTINFO to be reported to the user asynchronously.

```
CALL SSR$REG(SS_NQP);
```

Block the current user when no queue packets are available.

```
CALL SSR$RE(SS_QPA);
```

Unblock all users who were blocked for lack of a queue packet.

4.6 Request for Special Interrupt Control

Call form:

```
CALL NIO$SPINTCON(DCTX,INTADDR,EAINFO)ALTRET(label);
```

where:

DCTX is the logical device index for which special interrupt control is requested.

INTADDR is the procedure name (as a PL-6 entry pointer) to which control is to be transferred when a special interrupt occurs on this device. This address must be within the monitor's instruction segment. The special interrupt request remains in effect until it is cleared by a call to NIO$SPINTCON with EADDR=NULL.

EAINFO is a data word to be passed to the special interrupt end action routine.

The alternate return is taken if:

- DCTX is not a valid device table index.
- The device does not generate special interrupts.
- Special interrupts are already requested for the device.

4.7 Special Interrupt End Action Routine

Call form:

```
CALL INTADDR(DCTX,INTSTAT,EAINFO);
```

where:

INTADDR is the end-action routine specified in the call to NIO$SPINTCON.

DCTX is the device index of the interrupting device.

INTSTAT contains a logical status code indicating the type of special interrupt that occurred. See Section 3.2.

4.8 Clean Up All User I/O

Call form:

```
CALL NIS$ABIO(USR#);
```

where:

USR# is the user number.

Purpose:

Abort and clean up all I/O associated with the specified user.

5. ERROR RECOVERY AND LOGGING

5.1 Error Recovery

IOQ will perform all error recovery for normal I/O requests. The action taken depends on the device type and the status returned from the I/O operation.

Basically there are five distinct types of status which may occur during an I/O operation. These are listed in the following paragraphs with examples and action taken.

5.1.1 Normal Completion

Example:

Channel Ready

Action:

This the normal completion of an I/O request; a normal completion code is returned to the caller (i.e., to the DCB and/or end-action routine). This category includes status codes that indicate that the MPC (microprogrammed controller) firmware corrected an error.

5.1.2 Data/User Error

Example:

Mispunched ASCII card, blank tape read.

Action:

No recovery can be performed; a completion code indicating the type of error will be returned to the caller.

5.1.3 Read/Write/Transmission Error

Examples:

Tape parity error, disk checkword error, channel parity error.

5. ERROR RECOVERY AND LOGGING

5.1 Error Recovery

IOQ will perform all error recovery for normal I/O requests. The action taken depends on the device type and the status returned from the I/O operation.

Basically there are five distinct types of status which may occur during an I/O operation. These are listed in the following paragraphs with examples and action taken.

5.1.1 Normal Completion

Example:

Channel Ready

Action:

This the normal completion of an I/O request; a normal completion code is returned to the caller (i.e., to the DCB and/or end-action routine). This category includes status codes that indicate that the MPC (microprogrammed controller) firmware corrected an error.

5.1.2 Data/User Error

Example:

Mispunched ASCII card, blank type read.

Action:

No recovery can be performed; a completion code indicating the type of error will be returned to the caller.

5.1.3 Read/Write/Transmission Error

Examples:

Tape parity error, disk checkword error, channel parity error.

Action:

There are a variety of recovery techniques that may be used depending on the device type and the nature of the error. In some cases, the MPC for the device will retry the operation several times. If one of these retries is successful, a normal completion status will be returned to the CPU (see Section 5.1.1). If not, IOQ will perform recovery sequences which include additional retries. If one of these retries is successful, a normal completion code will be returned to the caller. If not, an unrecoverable error completion code will be returned to the caller.

5.1.4 Device Requires Operator Intervention

Examples:

Card reader hopper empty, line printer paper jam.

Action:

This type of status is divided into two sub-categories:

1) Those conditions that the operator can correct such as a card reader hopper empty condition. For these conditions, IOQ will notify the operator of the problem, wait for the special interrupt signifying that the device is ready, then retry the I/O operation.

2) Those conditions which may also require action by the caller, such as a line printer paper jam. For these conditions, IOQ will notify the operator and return an error completion code to the caller. The caller, usually a symbiont, can then conduct his own dialogue with the operator, reposition the print file, if necessary, and reissue the I/O request. (IOQ will wait for the special interrupt signifying that the device is ready before issuing the I/O request to the device.)

5.1.5 Software Errors

Example:

Invalid instruction sequence.

Action:

This type of status response to an I/O operation can only be caused by software errors in IOQ; it will cause a system crash.

5.2 Error Logging

In order to provide both immediate device error information as well as a record for later analysis, IOQ will keep a device error log as part of the system error log. It will contain two types of entries:

1) Every error status received will be logged, including those that indicate that the MPC successfully corrected an error. This type of entry will contain a time stamp, the external device ID, the IOM number, channel number, and MPC ID, the IDCW, the status word or doubleword, any other status information available, such as the extended status bytes generated by the unit record equipment, and the volume ID, if applicable.

2) Whenever a tape or disk volume ID is dismounted and periodically, if a volume remains mounted for an extended period of time, the device operation statistics will be read from the MPC and logged. This type of entry will include a time stamp, the external device ID, the MPC ID, the volume ID, and the device operation statistics.

6. TEST AND DIAGNOSTICS (T&D)

The T&D interface to IOQ will provide direct access to peripheral devices, MPCs, and IOMs (Input-Output Multiplexors) so that preventive and emergency maintenance can be performed on devices that have been partitioned out of the active system. This interface will be implemented so that a hardware diagnostic program can be a user job running in slave mode. The interface will allow the caller to specify:

- The device to be tested (external device ID),
- The path to that device (IOM number, channel number), and
- The PCW (Peripheral Control Word) and channel program to be issued to the device.

On completion of the request, the T&D interface will return the status doubleword and the contents of the channel's mailbox, whether in scratchpad or in memory, to the caller. No error recovery will be performed. If a special or fault interrupt occurs for the specified channel during the request, the status word from that interrupt will be returned to the caller, terminating the request.

The T&D interface will check all requests to ensure that system security is not compromised:

- Perpheral Device Access

  The requested device must be partitioned out. (T&D cannot be performed on a device which is available to CP-6.) The interface will set the device number field in each IDCW so that only device commands will be accepted by the hardware.

- MPC Access

  The requested MPC (i.e., all channels connected to that MPC) must be partitioned out. Device commands will be accepted only if they refer to devices which are partitioned out.

- IOM Access

  The T&D interface will allow access to the snapshot, wraparound, and scratchpad access channels. The PCWs will be checked to ensure that only partitioned-out channels are accessed.