ARIX Corporation

IOPM Boot Strategy

Rev 0.0

Norman C. Woo
March 5, 1989

1. Introduction

This document is a description how an IOPM based System 90 will perform its initial boot. The major objective is be flexible so that future new I/O Processor Modules and Device Boards can be easily adapted into existing systems.

2. Theory of Operation

The IOPM will automatically execute the power-on confidence tests, a series of EPROM based tests, every time it is reset. These tests will begin by checking the CPU, checksum the EPROM, test the RAM, the CSS bus interface, and then the interrupt mechanism. Once completed the appropriate LEDs will be set. The IOPM will interrupt the Service Module if the confidence tests did not detect any fault and will wait for further commands from the Service Module or any intelligent modules. In the event of finding any error, the IOPM will not interrupt the Service Module and won't respond to any command.

The IOPM receives the command from other intelligent modules in the system through a shared structure residing in the local RAM on the IOPM. Reguests for the IOPM to perform a task are made by writing a command into the structure and releasing it to the IOPM. Any originator, one who wants to be issue a command, must try to gain ownship of the structure before writing the structure. To become the owner of the structure, the originator first examines the state of the structure. If it is locked, or owned by others, the originator must wait until it is freed by the IOPM; otherwise, the originator can put a lock on the structure. Simply locking the structure does not mean possession. To ensure mutual exclusion, after putting on the lock an originator must also writes its unique id into the structure. The structure is considered belonging to an originator only when the owner id in the structure matches the originator's id. A non-owner must wait for the next time the structure becomes free.

The originator who owns the structure can now fill the structure with the required information for the IOPM to complete a command. With all the necessary information correctly placed, the originator can pass the ownership of the structure to the IOPM by changing the owner id and interrupts the IOPM, thus issuing the command to the IOPM.

The IOPM acknowledges the originator that the command was received by changing the status indicator in the structure. The IOPM will not change the status until it has completed the command. At the termination of the command, the status is changed to indicate the results are valid and ready for the originator; also the IOPM must interrupt the originator to signal the completion of the commnad.

After receiving the signal indicating the results are ready, the originator should retrieve all the results, change the originator id in the structure to that of the IOPM, and issue another interrupt to the IOPM. This allows IOPM to free the command block. The IOPM waits for this confirmation that the originator had received all the

results before freeing the command block for the next request. The IOPM frees the structure by clearing the lock and showing an idle status.

3. Command Block

This section describes the **Command Block**.

3.1 Command block structure

```
typedef struct cmd_blk {
        unsigned char     lock;
        unsigned char     owner;
        unsigned char     originator;
        unsigned char     status;
        unsigned short    command;
        unsigned short    ret_code;
        unsigned int      arg_cnt;
        unsigned int      *argv;
} CMD_BLK, *CMD_BLK_PTR;
```

3.2 Field Descriptions

lock is the key to this structure. A non-zero value means someone has ownership of this structure and only the owner can alter the content of the structure. A zero value in this field means the structure is freed by the IOPM. Any one can try to gain ownership by writing a non-zero value, but only the IOPM can free this structure.

owner shows who is the current owner of this structure. It is a two steps process for any one to become the true owner of the command block. After setting the **lock**, one must also write its unique system-slot-id into this field. Because of possible race condition, an verification of **owner** is a prerequisite before modifying the content. Any one sees other system-slot-id in **owner** will have to wait until the the structure is freed by examining the **lock**. To issue the command, simply put the IOPM system-slot-id into **owner**.

A system-slot-id is the 8 most significant bits(A28-A36) of a CSS system address that accesses that slot.

originator contains the system-slot-id of the one who issued the current command.

status indicates the current state of the IOPM.

command is the command code.

ret_code is the return code from executing the command. Zero for successful termination and non-zero values are error codes.

arg_cnt is the number of the arguments associated with this command. The number of arguments depends on the type of command issued.

argv is a pointer to first argument associated with this command. All arguments must be contiguous.

4. Command Descriptions

4.1 R_DEV

Read **DEV**ice command has the following arguments:

```
unsigned int    device;
unsigned int    block;
unsigned int    blk_cnt;
unsigned int    s_slot;
unsigned int    offset;
```

The IOPM will use the device driver resided in the EEPROM located on the attached device board to perform a Read Device operation with the given arguments.

Arguments descriptions:

device is the device number.

block is the starting block number.

blk_cnt is the number of block to be transferred.

s_slot is the system-slot-id of the destination.

offset is the offset within the destination module.

4.2 RSMXL

Read from System Memory and EXecute Locally. This command has the following arguments:

```
unsigned int    blk_cnt;
unsigned int    s_slot;
unsigned int    offset;
```

The IOPM will transfer an executable image from system memory into its local RAM and execute it.

Arguments descriptions:

blk_cnt is the number of block to be transferred.

s_slot is the system-slot-id of the source.

offset is the offset within the source module.

4.3 PROM_VER

PROM VERsion command has the following arguments:

unsigned int version;

The IOPM will return the currently installed IOPM PROM version number in the argument.

Arguments descriptions:

version is version number of the IOPM PROM.

4.4 EEPROM_VER

EEPROM VERsion command has the following arguments:

unsigned int version;

The IOPM will return the currently installed EEPROM version number in the argument.

Arguments descriptions:

version is version number of the EEPROM.

5. Status Descriptions

These are the valid status issued by the IOPM:

DONE means the command was successfully executed.

INVLD means an invalid command was received.

IDLE means IOPM is idle waiting for the next command.

6. Hardware Requirements

This is a list of the minimum hardware requirements for all IOPMs and Device Boards to maintain compatibility:

1. **Memory Location**. Since the IOPM receives its command as a structure in its local memory, all future IOPMs should reserve the same physical locations for memory.
2. **Interrupt Control Register**. The format and location of this register should not be change in any new IOPMs.
3. **EEPROM Location**. The format and location of the EEPROM on the device board should not be change.