SECTION VI

CONTROLLER INTERFACE

Most applications involving the hand controller require similar needs in decoding and debouncing those inputs. The operating system addresses those needs in one general purpose routine, POLLER. POLLER will decode and debounce either all or selected portions of the hand controller hardware and place the processed data in the Controller Data Area selected by the pointer in CONTROLLER_MAP.

Special applications may require non-standard decoding of the inputs available from the hardware; therefore, entry points to lower level routines are available.

There are four routines available to access controller inputs:

- POLLER
- DECODER
- CONT_SCAN
- UPDATE_SPINNER

6.1 Controller Data Area

The pointer in CONTROLLER_MAP points to the user-defined CRAM area which is accessed and/or modified when POLLER is called. Users define this address by placing the location of the 12 bytes of the CRAM Controller Data Area at cartridge location CONTROLLER_MAP. They are defined as follows:

+0	Player 1 enable	
+1	Player 2 enable	
+2	Fire button (left button)	Player 1
+3	Joystick	Player 1
+4	Spinner count (for interface modules)	Player 1
+5	Arm button (right button)	Player 1
+6	Keyboard	Player 1
+7	Fire button	Player 2
+8	Joystick	Player 2
+9	Spinner count	Player 2
+10	Arm button	Player 2
+11	Keyboard	Player 2

Player Enable (+0, +1):

Bit 7							Bit 0
	X	X			X		

Where bit = 1: Function enabled.

bit = 0: Function disabled.

X = Don't care

While functions are as follows:

Bit 7 = Controller Enable

Bit 4 = Keypad

Bit 3 = Arm Button

Bit 1 = Joystick

Bit 0 = Fire Button

Statu- of individual portions of the controller map area when enabled is described as follows:

Fire button:

Status = 040H, if fire button pressed

Status = 0H, if fire button not pressed

Joystick:

Status	Direction
01H	N
03H	NE
02H	E
06H	SE
04H	S
0CH	SW
08H	W
09H	NW

Spinner Switch:

SPIN_SW_CNT is added to the value for position offset.
(Ref to Sec. 6.5)

Arm Button:

Status = 0040H if arm button pressed

Status = 0000H if arm button not pressed

Keypad:

Value	Key
00H	0
01H	1
02H	2
03H	3
04H	4
05H	5
06H	6
07H	7
08H	8
09H	9
0AH	*
0BH	#

6.2 POLLER

Calling Sequence:

```
CALL  POLLER
```

Description:

Reads, decodes and debounces all active portions of both controllers. The results are placed in the Controller Data Area.

POLLER's debounce algorithm waits until it finds the data the same for two successive passes before it modifies the Controller Data Area. If a particular portion is disabled, then this routine will still be looking for the second occurrence upon re-enabling. Please note that the POLLER routine cannot interrupt itself.

Side Effects:

-Destroys all except alternate register pairs, does not destroy alternate AF pair.

- Zero's SPIN_SW_CNT if that portion of the controller is enabled. (See UPDATE_SPINNER).

Calls to other OS routines:

- CONT_SCAN

6.3 DECODER

Calling Sequence:

```
     LD    H, CNTRLR NO.
     LD    L, CNTRLR SEGMENT NO.
     CALL  DECODER
```

Description:

DECODER calls CONT_SCAN; decodes and returns as output according to the controller segment requested. Decoding uses the same format as the individual status bytes in Controller Data Area.

Parameters:

CNTRLR NO. — 0 = Player 1's controller only
1 = Player 2's controller only

CNTRLR SEGMENT NO. — The value found in segment number will decode these respective portions of the controller:

0 = Fire,Joystick, Spinner

1 = Arm, Keypad

OUTPUTS: IF SEGMENT CHOSEN WAS:

	Segment 0	Segment 1
Register H	Fire	Arm
Register L	Joystick	Keyboard
Register E	Spinner	

The decoded values are listed in the Controller Data Area.

Side Effects:

- Destroys AF, BC, DE and HL.

Calls to other OS routines:

- CONT_SCAN

6.4 CONT_SCAN

Calling Sequence:

```
     CALL  CONT_SCAN
```

Description:

Reads the actual ports to both controllers and places the data in an OS-defined CRAM area. These locations are labeled as S0_C0, S0_C1, S1_C0 and S1_C1.

Side Effects:

- Destroys AF.

6.5 UPDATE_SPINNER

Calling Sequence:

```
ORG   801EH
JP    UPDATE_SPINNER
```

Description:

For use with expansion modules only. Interrupt service routine which processes controller spinner switch interrupts (maskable). Decrements OS reserved byte SPIN_SWO_CNT for Controller No. 0 or SPIN_SW1_CNT for Controller No. 1 if spinner is going in one direction; increments byte if spinner is going in the other direction (Ref. Table 10-1).

NOTE: SPIN_SW_CNT is accessed and modified by both DECODER and POLLER if they are called.