

1
2
3 SECTION VI
4 CONTROLLER INTERFACE
5

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

13 Special applications may require non-standard decoding of the
14 inputs available from the hardware; therefore, entry points to
15 lower level routines are available.
16

17 There are four routines available to access controller inputs:
18

- 19 - POLLER
- 20 - DECODER
- 21 - CONT_SCAN
- 22 - UPDATE_SPINNER
- 23
- 24
- 25
- 26

1 6.1 Controller Data Area

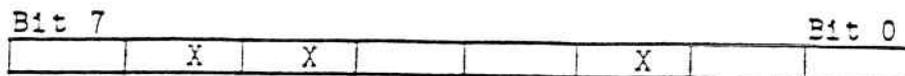
2
3 The pointer in CONTROLLER_MAP points to the user-defined
4 CRAM area which is accessed and/or modified when POLLER
5 is called. Users define this address by placing the
6 location of the 12 bytes of the CRAM Controller Data
7 Area at cartridge location CONTROLLER_MAP. They are
8 defined as follows:

9

10	+0	Player 1 enable	
11	+1	Player 2 enable	
12	+2	Fire button (left button)	Player 1
13	+3	Joystick	Player 1
14	+4	Spinner count (for interface modules)	Player 1
15	+5	Arm button (right button)	Player 1
16	+6	Keyboard	Player 1
17	+7	Fire button	Player 2
18	+8	Joystick	Player 2
19	+9	Spinner count	Player 2
20	+10	Arm button	Player 2
21	+11	Keyboard	Player 2

22
23
24
25
26

1
2 Player Enable (+0, +1):



5 Where bit = 1: Function enabled.

6 bit = 0: Function disabled.

7 X = Don't care

8
9 While functions are as follows:

10 Bit 7 = Controller Enable

11 Bit 4 = Keypad

12 Bit 3 = Arm Button

13 Bit 1 = Joystick

14 Bit 0 = Fire Button

15
16
17 Status of individual portions of the controller map area
18 when enabled is described as follows:

19
20 Fire button:

21 Status = 040H, if fire button pressed

22 Status = 0H, if fire button not pressed

23
24
25
26

1 Joystick:

2

3

4	<u>Status</u>	<u>Direction</u>
5	01H	N
6	03H	NE
7	02H	E
8	06H	SE
9	04H	S
10	0CH	SW
11	08H	W
12	09H	NW

13 Spinner Switch:

14

15 SPIN_SW_CNT is added to the value for position offset.

16 (Ref to Sec. 6.5)

17

18 Arm Button:

19

20 Status = 0040H if arm button pressed

21 Status = 0000H if arm button not pressed

22

23

24

25

26

Keypad:

	<u>Value</u>	<u>Key</u>
1		
2		
3		
4	00H	0
5	01H	1
6	02H	2
7	03H	3
8	04H	4
9	05H	5
10	06H	6
11	07H	7
12	08H	8
13	09H	9
14	0AH	*
15	0BH	#
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		

1
2
3 6.2 POLLER
4

5 Calling Sequence:

6
7 CALL POLLER
8

9 Description:

10
11 Reads, decodes and debounces all active portions of both
12 controllers. The results are placed in the Controller
13 Data Area.

14
15 POLLER's debounce algorithm waits until it finds the
16 data the same for two successive passes before it
17 modifies the Controller Data Area. If a particular
18 portion is disabled, then this routine will still be
19 looking for the second occurrence upon re-enabling.
20 Please note that the POLLER routine cannot interrupt
21 itself.
22
23
24
25
26

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

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

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

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

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 SO_CO, SO_C1, SI_CO and SI_C1.

Side Effects:

- Destroys AF.

1
2
3 6.5 UPDATE_SPINNER
4

5 Calling Sequence:

6
7 ORG 801EH
8 JP UPDATE_SPINNER
9

10 Description:

11
12 For use with expansion modules only. Interrupt service
13 routine which processes controller spinner switch
14 interrupts (maskable). Decrements OS reserved byte
15 SPIN_SWO_CNT for Controller No. 0 or SPIN_SW1_CNT for
16 Controller No. 1 if spinner is going in one direction;
17 increments byte if spinner is going in the other
18 direction (Ref. Table 10-1).
19

20 NOTE: SPIN_SW_CNT is accessed and modified by both
21 DECODER and POLLER if they are called.
22
23
24
25
26

