Automatic Interactive Control Reference Sheet for Block Computer Programmers

Terms:

Master Controller: The Hornby Zero One master control unit. The master controller sends commands and data through the tracks to the engines and block computers.

Block Computer: A student-built single board computer responsible for one block of the track layout.

Dispatch computer: The IBM PC/AT which monitors the track layout.

Purpose of Interactive Control:

- Fully automatic operation of projects. The dispatch computer takes the place of an operator typing on the Hornby keypad. Block computers can set train speed and direction through the dispatch computer and master controller.
- Communication between block computers. Block computer can broadcast messages to other blocks through the dispatch computer, and receive messages from the master controller.

Note: You can not count on a quick response from the dispatch computer.

Block Computer Program Startup:

All block computer programs should include version 2.0 of the Lecky initialization and interrupt service routines. The block computer program start up sequence must include the following code, which is to be run once.

```
        ORG     $F000           ; origin of code set to bottom of ROM; usually $F000.
CLD --> SEI
        LDX     #$FF
        TXS                     ; set stack pointer to $FF
        LDA     #02             ; load A with block computer id, a number between 1 and 8, 2 in
this example.
        STA     BLKID           ; store id value to the Lecky "block id" mailbox.
        LDA     #00             ; clear A
        STA     DFLAG           ; store to the Lecky "data to send" mailbox
        JSR     INIT            ; run the Lecky initialization routine.
```

Sending and Receiving Data from a Block Computer :

Block computers communicate by sending a one byte **request code** and one byte of **request data** to the dispatch computer. The dispatch computer uses the **request code** and data to determine the response that it sends to the master controller. This response can change the speed and direction of a train, and can also send one byte of **return data** back to the block computer.

The request code and data can be any one byte values; that is, decimal 0 through 255. However, for each transmission, either the request code or the data must be non-zero.

The return data sent to a block by the master controller can only take decimal values 01 through 09, or 80 through 89.

The block computer programmer must run the mpu program on the dispatch computer, and enter a response for each **request code** that the block computer will send. See Tad Coburn's report (1987) for details of dispatch computer response strings. The response string file name for the block should be included with the block's instructions for use.

Lecky Communication Mailboxes:

From the block computer program viewpoint, communication merely involves setting and reading Lecky routine mailboxes:

- **WDATAH** holds the **request code.**
- **WDATAL** holds the **request data.**
- **DFLAG** is a handshake mailbox. It is set to #$FF to indicate to the interrupt service routine that WDATAH and WDATAL contain valid data for the dispatch computer. The interrupt service routine clears DFLAG to #00 after the data has been sent.
- **RDATA** holds **return data** sent from the master controller to the block computer. Since valid return data is in the range 01-09 or 80-89, a student program can clear RDATA to #00 and watch for it to be set.

ORIGINAL 237

Handshaking:

A block computer program uses the RDATA and DFLAG mailboxes to determine when requests have been transmitted. When the block computer needs a specific action to be completed, it should wait for an **acknowledgement** message from the dispatch computer. In these cases, RDATA is used for handshaking. If the block computer is sending a message which is not acknowledged by the dispatch computer, it should use DFLAG for handshaking.

Handshaking with RDATA (ACKNOWLEDGED REQUEST) :

1) Store a **request code** to WDATAH
2) Store the **request code** to the display.
3) Store **request data** to WDATAL.
4) Clear RDATA to 00.
5) Set DFLAG to #$FF, to indicate to the interrupt service routine that there is data to send.
6) Loop until RDATA is set.
7) Test that RDATA contains the expected acknowledgement value.
   (Continue program. Return to step **1** for another acknowledged request.)

Step two makes the **request code** visible on the block's display. This is necessary so that the block functions properly in **manual mode**, with a human operator typing commands at the Hornby. RDATA handshaking works for either **automatic** or **manual mode.**

Handshaking with DFLAG (NON-ACKNOWLEDGED REQUEST) :

1) Loop until DFLAG = 00. This indicates that any pending **request data** has been sent.
2) Store the **request code** to WDATAH.
3) Store the **request code** to the display.
4) Store **request data** to WDATAL.
5) Set DFLAG to #$FF, to indicate to the interrupt service routine that there is data to send.
   (Continue program. Return to step **1** for another non-acknowledged request.)

When the dispatch computer commands the block computer to send its **request data**, DFLAG is cleared. Thus, in manual mode, the operator must enter the "send data" command to respond to non-acknowledged request. This command is [*block ID*] 0 -> on the Hornby keypad. This should be noted on the instructions for use of the block.

Switching between handshake protocols:

If a program has been using RDATA handshaking, and all of it's previous requests have been acknowledged, it may make an non-acknowledged request without waiting for DFLAG to clear. The acknowledgement means the previous request was completed, yet DFLAG may still be set if the block is being run in **manual mode.**

However, if a block's previous request used DFLAG handshaking, the block computer program must wait for DFLAG to clear before beginning any other request. Otherwise, the program may change WDATAH and WDATAL before the previous request has been sent.