List Processor Case Study #1.

A typical example of the use of various features of the LP1341/1342.

1. Introduction

In this example, an ADC/TDC readout system is considered, where variable amounts of data will be available from each module. This is typical of many event-driven data acquisition problems.

2. The System

The CAMAC system consists of a crate controller connected to a host computer, usually a PC, the list processor and a set of input modules. One of the input modules, normally the last to complete its conversions, will be used to trigger the list processor through its LAM output. For simplicity, the ADC/TDC units will be considered to be in slots 1 to 5, and the list processor in station 16.

3. The variable data problem

Whenever data acquisition modules are capable of producing a varying amount of data, the system designer is faced with the problem of interpretation of the assembled block of data, since it is difficult to know where an event record begins and ends. To aid this, the list processor can read its own instruction pointer, which will always give a fixed 24-bit data number with the top 8 bits all 1's. This is traditionally known as a 'tag' and can be inserted in the data record at any desired point to separate records or parts of records. Our example will use this tag twice.

4. The sequence

LAM trigger -> tag read -> read ADC1 until all data read -> read ADC2 until all data read -> tag read -> read TDC1 until all data read -> read TDC2 until all data read -> read ADC3 until all data read -> jump to start and await trigger.

5. The instruction map (all values in HEX)

Address	Contents		Comments
00	C100		Bootstrap table
01	C100		all jump to offset 100
02	C100		
03	C100		
04	C100		
05	C100		
06	C100		
…			
100	2001	**TAG**	N(16) A(0) F(1) read list processor inst. Pointer
101	4202		N(1) A(0) F(2) Q ignore
102	4222		N(1) A(1) F(2) Q ignore
103	4242		N(1) A(2) F(2) Q ignore
104	4262		N(1) A(3) F(2) Q ignore – Read ADC1 channels
105	4282		N(1) A(4) F(2) Q ignore
106	42A2		N(1) A(5) F(2) Q ignore

107	42C2		N(1) A(6) F(2) Q ignore
108	42E2		N(1) A(7) F(2) Q ignore
109	4402		N(2) A(0) F(2) Q ignore
110	4422		N(2) A(1) F(2) Q ignore
111	4442		N(2) A(2) F(2) Q ignore
112	4462		N(2) A(3) F(2) Q ignore – Read ADC2 channels
113	4482		N(2) A(4) F(2) Q ignore
114	44A2		N(2) A(5) F(2) Q ignore
115	44C2		N(2) A(6) F(2) Q ignore
116	44E2		N(2) A(7) F(2) Q ignore
117	2001	**TAG**	N(16) A(0) F(1) read list processor inst. Pointer
118	4602		N(3) A(0) F(2) Q ignore – read TDC data
119	C921		Jump if No Q to offset 121
120	C118		Jump back to offset 118 and continue
121	4802		N(4) A(0) F(2) Q ignore
122	C924		Jump if No Q to offset 124
123	C121		Jump back to offset 121 and continue
124	4A02		N(5) A(0) F(2) Q ignore
125	4A22		N(5) A(1) F(2) Q ignore
126	4A42		N(5) A(2) F(2) Q ignore
127	4A62		N(5) A(3) F(2) Q ignore
128	4A82		N(5) A(4) F(2) Q ignore
129	4AA2		N(5) A(5) F(2) Q ignore
130	4AC2		N(5) A(6) F(2) Q ignore
131	4AE2		N(5) A(7) F(2) Q ignore
132	E000		Jump to zero and await trigger.

6. Points to notice about this list of commands:

We are dealing with two very different kinds of module here; the ADC clearly has 8 channels of data any of which may have converted, some may not, so we have to explicitly read them all out and use the presence of Q='1' to tell the list processor whether to store the data or not. The second type, the TDC, has just one readout command which acts like a FIFO and we keep reading it until it gives No Q.

Notice also that you will also need to include commands in the list to get the modules ready for the next event, either to reset them and/or re-enable them. You also need to make sure that the module which caused the trigger by asserting its LAM has been cleared to remove this signal, otherwise as soon as this list completes, it will get restarted straight away.

Peter Marshall, December 2000-12-05. peter@hytec-electronics.co.uk