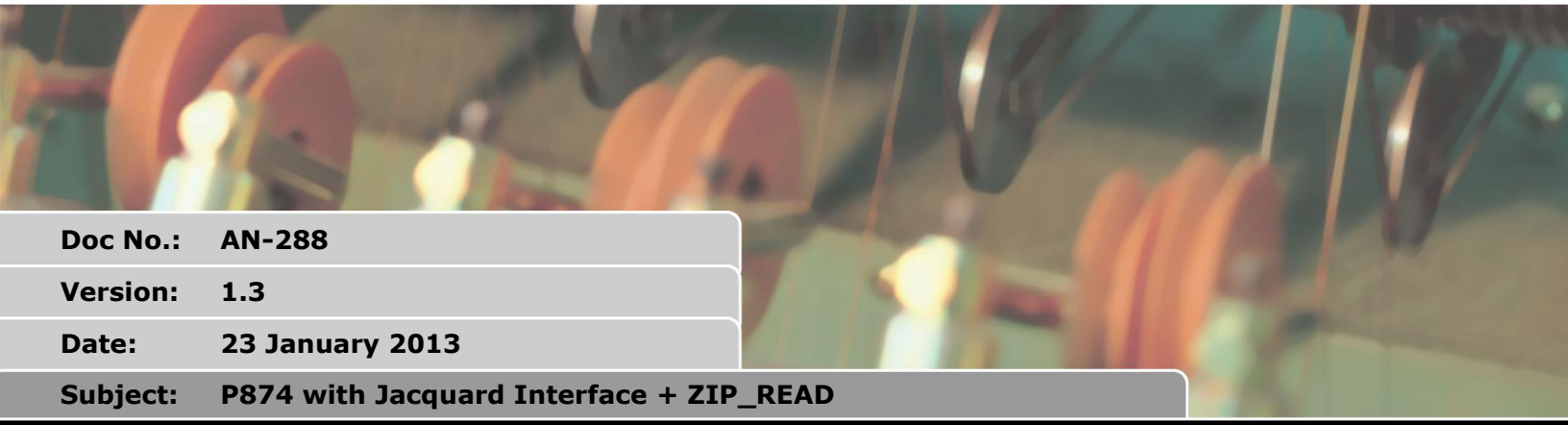
Shannon Way,
Tewkesbury,
Gloucestershire. GL20 8ND
United Kingdom
Tel: +44 (0)1684 292 333
Fax: +44 (0)1684 297 929

187 Northpointe Blvd,
Suite 105
Freeport, PA 16229
United States of America
Tel: +1 724-540-5018
Fax: +1 724-540-5098

Tomson Centre
118 Zhang Yang Rd., B1701
Pudong New Area, Shanghai,
Postal code: 200122
CHINA
Tel/Fax: +86 21 587 97659

SCMC House
16/6 Vishal Nagar
Pimpale Nilakh, Wakad, Pune
PIN 411027
INDIA
Tel: +91 206 811 4902



Doc No.: AN-288

Version: 1.3

Date: 23 January 2013

Subject: P874 with Jacquard Interface + ZIP_READ

APPLICATION NOTE

1. Scope

This document provides information on using a P874 with a custom programmed FPGA allowing control of a Jacquard distribution board via SPI over an RS422 physical layer. Software enhancements are also described which allow a data file to be transmitted to the controller in compressed (Zipped) format and stored on a Motion Coordinator in this format.

2. Overview

The key differences to a standard P874 are:

- Axis 0 and Axis 4 arranged provide a bi-directional SPI interface
- Support for maximum of 24576 outputs via SPI (Larger numbers will require larger FPGA to be fitted to the P874 PCB)
- The other 6 axes can be used for servo or stepper control
- PSwitch functionality as per standard P874
- 2 hardware pulses counters with additional gating inputs
- Support for absolute axes removed to make space in the FPGA

3. Connection Pinout

P874			Jacquard Board	
Axis	Channel	Pin Number	SPI Signal	Pin Number
0	A	1	CLK-IN+	5,6
0	/A	2	CLK-IN-	7,8
0	B	3	MOSI+	9,10
0	/B	4	MOSI-	11,12
0	Z	6	MISO+	18,19
0	/Z	7	MISO-	20,21
4	A	9	OE-IN+	1,2
4	/A	10	OE-IN-	3,4
4	B	11	LOCK+	16,17
4	/B	12	LOCK-	14,15

4	Z	13	EN+	22,23
4	/Z	14	EN-	24,25
-	0V	5,15	GND	13,26

4. Programming

JACQUARD(function, slot [, parameters])

Function 0

Configuration - Defines where the pattern is stored in the vr variables - its base_address and number of values.

JACQUARD(0, slot, base_addr, length [,interval])

(Interval is no longer used, we originally wrote out the pattern periodically, now we only write it out when triggered by Function 1 and/or 2)

Function 1

Trigger - the pattern held within the vr variables is written out to the FPGA FIFO.

JACQUARD(1, slot)

Function 2

SD card operation - Import pattern from SD card and write out to FPGA FIFO.

Parameters are function, slot, then same as per StickReadVr(). The pattern is stored in the vrs and hence can be written out again using Function 1 if required.

JACQUARD(2, slot, SD filename, vr_index, format)

Read Status : JACQUARD(10,slot [,vr_index])

Read Base address : JACQUARD(11, slot [,vr_index])

Read Length : JACQUARD(12, slot [,vr_index])

Read Timeout : JACQUARD(13, slot [,vr_index])

The above read functions allow the data set with functions 0 and 2 to be read back.

Read FifoStatus : JACQUARD(14, slot [,vr_index])

Reads hardware FIFO status:

Bit(0) = FIFO empty flag (1 = empty). Check FIFO empty before writing a new data.

Bit(1) = FIFO full flag (1 = full)

Set SPI Baudrate : JACQUARD(15,slot,datarate)

Division of 20MHz clock to generate SPI clock. Divide by n+1, i.e. default 9 => divide by 10 to give 2M baudrate, or 1MHz SPI clock out.

Set Pulse Count Control Register : JACQUARD(16,slot,value)

Value is directly written to the register defined below:

Pulse Count Control Register:

Bit(s)	*Description*	*Reset State*	*Function*
0	Enable Count 0	0	0 = Pulse Counter Disabled 1 = Pulse Counter Enabled
1	Reset Count 0	0	0 = Pulse Counter Running 1 = Pulse Counter Reset
2	Counter wrap 0	0	0 = Counter automatically wraps from max count back to 0 1 = Counter stops when max count reached
3	Enable input gate 0	0	0 = Counter input -> Input(0) 1 = Counter input -> Input(0) AND Input(4)
4-7	Unused	0000	
8	Enable Count 1	0	0 = Pulse Counter Disabled 1 = Pulse Counter Enabled
9	Reset Count 1	0	0 = Pulse Counter Running 1 = Pulse Counter Reset
10	Counter wrap 1	0	0 = Counter automatically wraps from max count back to 0 1 = Counter stops when max count reached
11	Enable input gate 1	0	0 = Counter input -> Input(1) 1 = Counter input -> Input(1) AND Input(5)
12-15	Unused	0000	
16-23	Filter Clock Divisor	$31	Division of 20MHz clock to generate filter clock. Divide by n+1, i.e. default 49 => divide by 50 to give 400KHz filter clock out. The filter has 4 clock stages so default minimum pulse width of 10us.
24-31	Unused	$00	

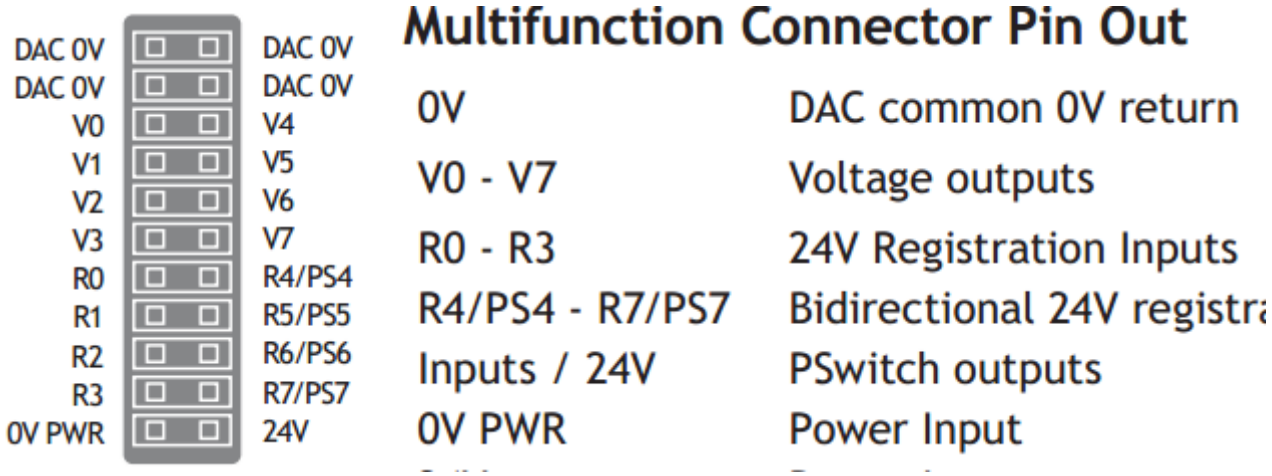
Read Counter 0 : JACQUARD(17,slot[,vr_index])

Read the number of counts in counter 0.

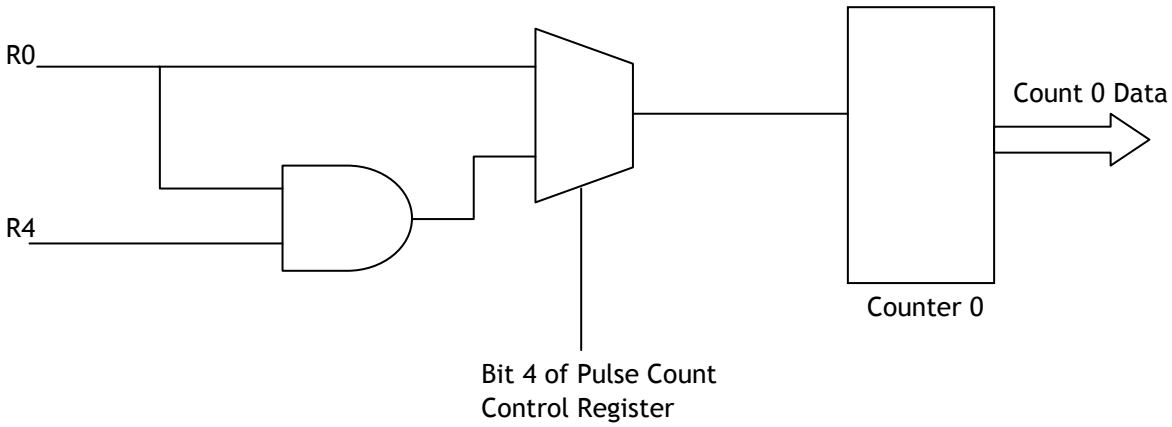
Read Counter 1 : JACQUARD(18,slot[,vr_index])

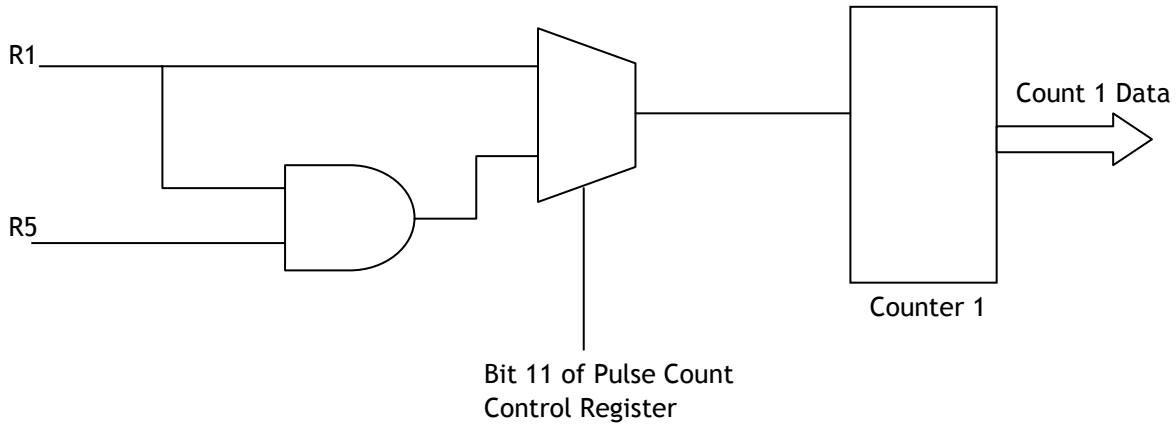
Read the number of counts in counter 1.

5. Counter Inputs



The registration inputs R0 and R1 can be used as high speed counters. Inputs R4 and R5 can be used to gate the counter inputs to prevent counting during some of the machine cycle.





The gating inputs R4 and R5 can be driven from a machine input or to provide a gating function based

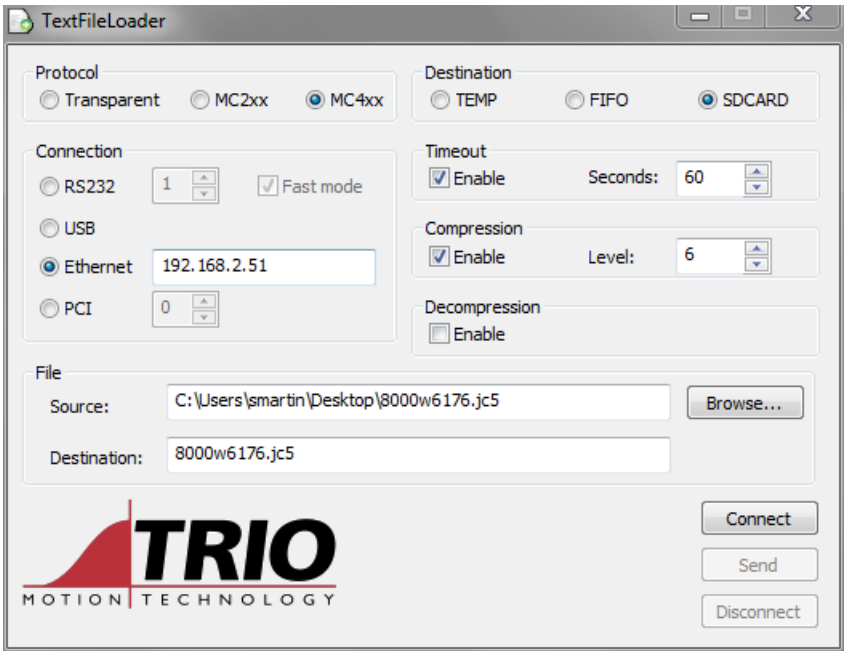
on position a PSWITCH output can be used on the controller. The output driven by the PSWITCH must be connected to R4 or R5.

6. SD Card File Transfers

Open the TextFileLoader. You will need to set the following options:

Protocol = MC4xx
Connection = Ethernet and the IP address of your controller
Destination = SD card
Compression = Enable. Compression level is typically set to 6.
Decompression = Disable. Very important as the file is stored compressed on the SD card !

One you have set the options then select the source file and transfer. This is how the text file loader will appear:



7. Reading Compressed SD Card Files

The software command “ZIP_READ” allows compressed files to be read progressively as the machine works through the data:

- Step A: Read compressed data file once from SD card into a special RAM buffer.
- Step B: If the file has a header skip

ZIP_READ(0,"filename") – Transfers a compressed SD card file into a special RAM buffer. Returns 0 – file read fails, 1 – partial read file, 2 – whole compressed file transferred complete OK into RAM

buffer. The RAM buffer is currently 1M byte for testing but will be set to 8M bytes. This should allow for uncompressed files of over 100M bytes.

ZIP_READ(1) – Allows a partially read file to be closed if ZIP_READ(0,"filename") returns 1.

ZIP_READ(2, format, destination, start, length) - Reads "length" values from the decompressed data and stores them in the VR/TABLE starting at "start". The data is assumed to be formatted as per the "format" parameter. Valid values for format are:

"format"	File Data format
0	8 bit integer
1	16 bit integer little endian
2	16 bit integer big endian
3	32 bit integer little endian
4	32 bit integer big endian
5	64 bit integer little endian
6	64 bit integer big endian

"destination"	Controller memory type
0	TABLE
1	VR

ZIP_READ(3,nbytes) - skip "nbytes" bytes in the file. This allows file headers to be skipped.

v=ZIP_READ(4,value)

This allows the BASIC program to check the current position in the file, the file length etc. "value" can be:

"value"	Returns
0	compressed buffer offset
1	compressed buffer length
2	compressed file offset
3	uncompressed buffer offset
4	uncompressed buffer length
5	uncompressed file offset

ZIP_READ(5) - This allows the file offset pointers to be returned to the start of the file.

Returns 0,1 or 2 similar to ZIP_READ(0,…)

Example file read:

Below is a sample file 321b.JC5 This is a part of the 321.JC5 file. The file has a header of 52 bytes then blocks of 180 bytes of data.

```
00000000h: 81 FE 01 01 00 00 00 02 00 00 00 01 00 00 00 04 ; þ..............
00000010h: 33 32 31 00 00 00 00 02 00 00 00 04 00 00 00 20 ; 321............
00000020h: 00 00 00 01 00 00 00 64 00 00 00 08 00 00 00 01 ; .......d........
00000030h: 00 00 05 A0 80 80 80 80 FF 7F FF FF FF FF FF FF ; ... €€€€ÿ ÿÿÿÿÿÿ
00000040h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000050h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000060h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000070h: FF FF FF FF FF FF FF FF FF 7F FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿ ÿÿÿÿÿÿ
00000080h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000090h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000a0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000b0h: FF FF FF FF FF FF FF FF FF 7F FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿ ÿÿÿÿÿÿ
000000c0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000d0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000e0h: FF FF FF FF FF FF FF FF 40 40 40 40 FF FF FF 7F ; ÿÿÿÿÿÿÿÿ@@@@ÿÿÿ
000000f0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000100h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000110h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000120h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 7F ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000130h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000140h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000150h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000160h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 7F ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000170h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000180h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000190h: FF FF FF FF FF FF FF FF FF FF FF FF 20 20 20 20 ; ÿÿÿÿÿÿÿÿÿÿÿÿ
000001a0h: FF FF FF FF FF 7F FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿ ÿÿÿÿÿÿÿÿÿÿ
000001b0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000001c0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000001d0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000001e0h: FF FF FF FF FF 7F FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿ ÿÿÿÿÿÿÿÿÿÿ
000001f0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

```
00000200h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000210h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000220h: FF FF FF FF FF 7F FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿ ÿÿÿÿÿÿÿÿÿÿ
00000230h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000240h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000250h: 10 10 10 10 FF FF FF FF FF FF FF 7F FF FF FF FF ; ....ÿÿÿÿÿÿÿ ÿÿÿÿ
00000260h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000270h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000280h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000290h: FF FF FF FF FF FF FF FF FF FF FF 7F FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿ ÿÿÿÿ
000002a0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000002b0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000002c0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000002d0h: FF FF FF FF FF FF FF FF FF FF FF 7F FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿ ÿÿÿÿ
000002e0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000002f0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000300h: FF FF FF FF                                     ; ÿÿÿÿ
```

```
DIM filename AS STRING(20)

filename = "321b.jc5"

IF ZIP_READ(0,filename)=2 THEN
    PRINT "File transferred into buffer OK"
ELSE
    PRINT "File read failed"
    STOP
ENDIF

ZIP_READ(3,52) ' Skip Header

block=90

REPEAT
    n=ZIP_READ(2,2,1,1000,block) ' Read 90 x 16 bit values big endian

    ' block of data can be transferred using JACQUARD command here

UNTIL n < block

' machine file read can repeat with ZIP_READ(5) here
```