Key Design Features

- Synthesizable, technology independent IP Core for FPGA and ASIC
- Supplied as human readable VHDL (or Verilog) source code
- 24-bit RGB video support with option for YCbCr video formats if required
- Generates clean and progressive video output without combing or tearing
- Reduced softening and sawtooth artefacts
- Supports three different de-interlacing modes including: Interpolated BOB, ELA (Edge-based Line-Average) and a customized version of LCI (Low-Complexity Interpolation)
- Supports all interlaced video formats up to 4096 x 4096 pixels in resolution. Examples include: 480i, 576i, 1080i etc.
- Output is one frame per interlaced field
- Fully pipelined architecture with simple flow-control
- No frame buffer required
- Supports 200MHz+ operation on basic FPGA devices

Applications

- High-quality video de-interlacing without the overhead of a frame buffer
- Conversion of 'legacy' SDTV formats to HDTV video formats
- Generating progressive RGB video via inexpensive PAL/NTSC decoder chips
- Digital TV set-top boxes and home media solutions

Generic Parameters

Generic name	*Description*	*Type*	*Valid range*
deint_mode	De-interlacing mode	integer	0: BOB 1: ELA 2: LCI 3: MIX (option)
frame_rate	Output frame rate	integer	0: min 1: max
line_width	Width of linestores in pixels	integer	2^4 < pixels < 2^{12}
log2_line_width	Log2 of linestore width	integer	log2(line_width)
field_polarity	Polarity of the 'pixin_field' input when the field is even	std_logic	0: even field signified by '0' 1: even field signified by '1'

Block Diagram

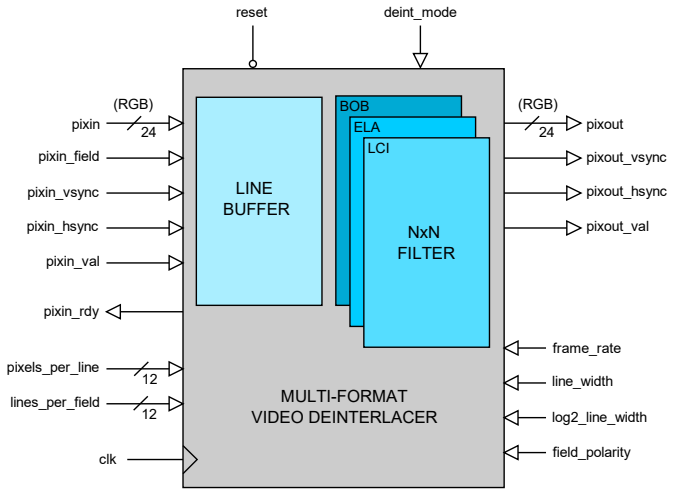

Figure 1: Video deinterlacer architecture

Pin-out Description

Pin name	*I/O*	*Description*	*Active state*
clk	in	Synchronous clock	rising edge
reset	in	Asynchronous reset	low
pixels_per_line [11:0]	in	Number of pixels per input line	data
lines_per_field [11:0]	in	Number of lines per field	data
pixin [23:0]	in	24-bit RGB pixel in	data
pixin_field	in	Input field number	data
pixin_vsync	in	Vertical sync in (Coincident with first pixel of a new input field)	high
pixin_hsync	in	Horizontal sync in (Coincident with first pixel of a new input line)	high
pixin_val	in	Input pixel valid	high
pixin_rdy	out	Ready to accept input pixel (handshake signal)	high
pixout [23:0]	out	24-bit pixel out	data
pixout_vsync	out	Vertical sync out (Coincident with first pixel of a new output frame)	high
pixout_hsync	out	Horizontal sync out (Coincident with first pixel of a new output line)	high
pixout_val	out	Output pixel valid	high

General Description

The DEINTERLACER IP Core is a high quality 24-bit RGB video deinterlacer capable of generating progressive output video at up to 4096x4096 pixels in resolution. The design is fully customizable, supporting any desired interlaced video format.

The deinterlacer allows for three possible filter algorithms - either BOB, ELA or LCI. All three methods are 'intra-field' methods that perform spatial filtering within the same field. For this reason, the output video is not subject to combing or tearing which is characteristic of a traditional 'weave' approach.

Each algorithm has it relative merits in terms of image quality and hardware complexity. In particular, the enhanced LCI algorithm provides excellent all-round performance with reduced image softening and crisp clean edges.

Pixels flow into the module in accordance with the valid-ready pipeline protocol. The pixel, sync flags and field number are transferred into the deinterlacer on a rising clock-edge when *pixin_val* and *pixin_rdy* are both active high. At the output interface, pixels and syncs are valid on a rising clock-edge when *pixout_val* is high.

The basic architecture of the deinterlacer is shown in Figure 1. Input lines are buffered and organised spatially before being filtered according to the chosen algorithm. Each input field is converted to a single output frame with twice the number of lines per field.

Output Frame rate

When the generic parameter *frame_rate* is set to '1' then the output frame rate is equal to the input field rate. When set to '0', the output frame rate is half the field rate. For example, consider an interlaced video input at 50 fields/s. When the frame rate is set to '1' then the output video will be generated at 50 frames/s. Conversely, when the frame rate is set to '0', then output video will be generated at 25 frames/s.

At half the frame rate, only the even field will generate a complete output frame, and the odd field will be discarded. The polarity of the even field is controlled by the generic parameter *frame_polarity*.

Pixels per line and lines per field

The input signals *pixels_per_line* and *lines_per_field* define the format of the interlaced video input. As an example, these values would be set as '720' and '240' if the input video format was digitized NTSC at 720x480 resolution (480i). These values may be modified during normal operation of the deinterlacer. Any changes must be followed by a system reset.

The width of the linestores must be sufficient to hold a complete line of interlaced video. The width should be set to the nearest power of 2. For example, if *pixels_per_line* is set to '720', then *line_width* should be set to '1024' and *log2_line_width* should be set to '10'.

Flow control

Pixels flow into the deinterlacer in accordance with the valid-ready pipeline protocol[1]. At the input interface, the signal *pixin_hsync* is coincident with the first pixel of a new line. The signals *pixin_vsync* and *pixin_field* are coincident with the first pixel of a new field. All input signals are qualified by the *pixin_val* signal being asserted high.

1 See Zipcores application note: app_note_zc001.pdf for more examples of how to use the valid-ready pipeline protocol

In addition, the input interface uses the handshake signal *pixin_rdy*. When the module asserts *pixin_rdy* low, then all input signals must be stalled until *pixin_rdy* is asserted high again. On the output side, pixels and syncs are valid when *pixout_val* is asserted high.

On receipt of the first valid vsync after reset, the deinterlacing operation begins and output lines are generated in accordance with the chosen filter algorithm. The deinterlacer will generate two output lines for every input line while the input field is active. Due to the uneven ratio of input to output lines then, on average, *pixin_rdy* will have a 50% duty cycle. In order to maintain maximum pixel throughput without stalling, the deinterlacer should be clocked at at least double the input pixel rate. A typical arrangement is shown in Figure 2 below:

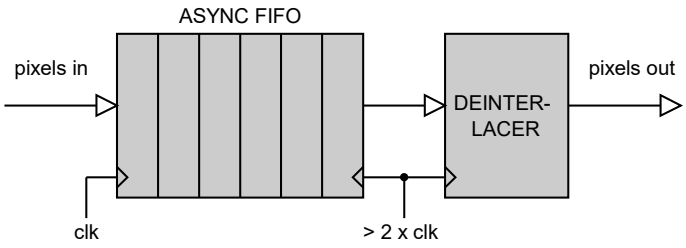

Figure 2: Deinterlacer clocking arrangement for maximum efficiency

Deinterlacing filter algorithm

The generic parameter *deint_mode* selects one of three possible deinterlacing filter algorithms. These are BOB, ELA or LCI. The choice of algorithm will determine the quality of the resulting output video as well as the size and complexity of the hardware implementation. The following table outlines the basic characteristics of each method. For empirical test results for each mode, please refer to the performance section of this document.

Deint_mode	***Description and properties***
0: BOB	Traditional 'bob' approach. Bilinear interpolation is used between adjacent lines to give a smooth graduated image. Method works very well with natural images. Sawtooth artifacts may be present if the image contains sharp lines and edges. Tends to soften image slightly. Results in a very small and fast hardware implementation suitable for lower-end applications.
1: ELA	This method uses a filter window to determine edge-vectors within a 3x3 block. Interpolation is performed according to the calculated vectors. Generates crisp and sharp output video. Some minor pixel displacements may be evident when edges are estimated incorrectly. Results in a medium hardware implementation size.
2: LCI	Most complex algorithm. Uses a 5x5 filter window and calculates more edge-vectors than ELA. Interpolation is performed in more directions and with more pixels. Offers balanced contrast without too much softening. Overall video quality is consistently better than BOB or ELA. Results in the largest hardware implementation size.

Figure 3 demonstrates the effect of each algorithm on a simple white diagonal line. Image (a) represents the original source image (without interlacing). Image (b) is the even field after interlacing. Images (c), (d) and (e) represent the result after deinterlacing the even field using the three filter algorithms.

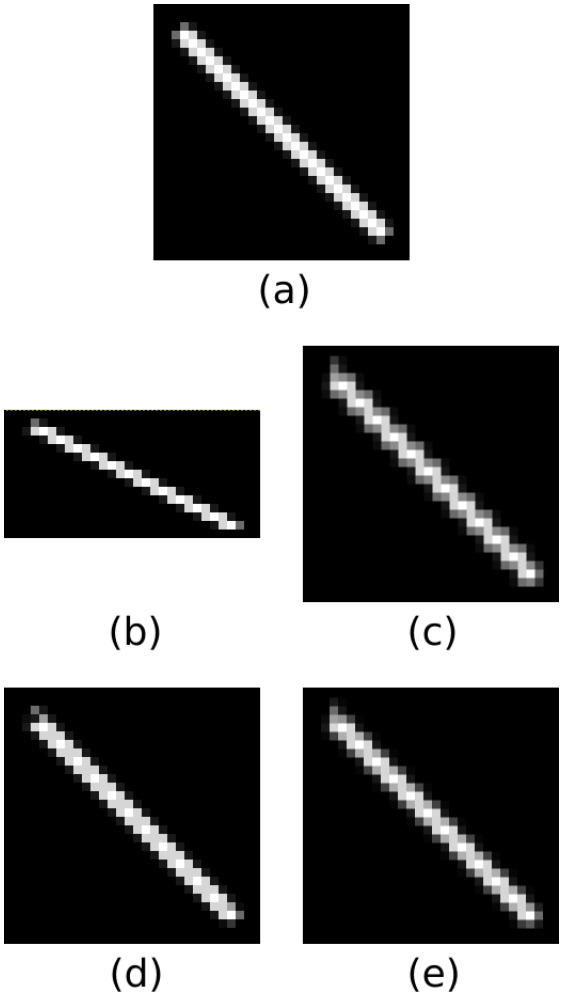

Figure 3: Visual effect of each filter algorithm: (a) Original image, (b) Interlaced field, (c) BOB, (d) ELA, (e) LCI

Functional Timing

Figure 4 shows the signalling at the input to the deinterlacer at the start of a new field. The first line of a new field begins with *pixin_vsync* and *pixin_hsync* asserted high together with the first pixel. Note that the signals *pixin, pixin_vsync* and *pixin_hsync* are only valid if *pixin_val* is also asserted high. In addition, the diagram shows what happens when *pixin_rdy* is de-asserted. In this case, the pipeline is stalled and the upstream interface must hold-off before further pixels are processed.

The signal *pixin_field* is a flag which identifies whether the input field is odd or even. This flag is only sampled at the start of a new field when *pixin_vsync* and *pixin_val* are high.

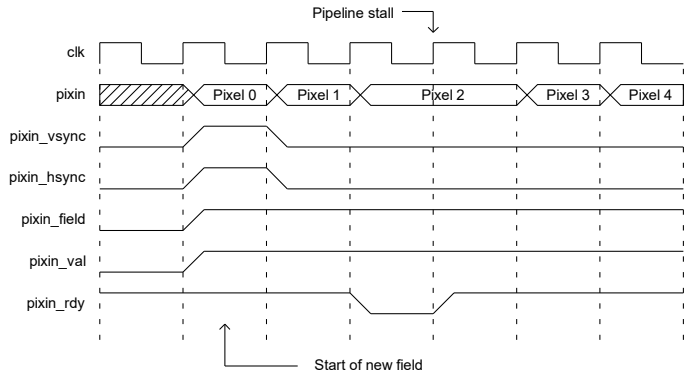

Figure 4: Deinterlacer input timing at the start of a new field

Figure 5 shows the signalling at the output of the deinterlacer. The output uses exactly the same protocol as the input with the exception that there is no 'ready' handshake signal. Note also that there is no 'field' flag as the output video is fully progressive. The timing diagram shows the output timing for a complete line. Outputs are only valid if *pixout_val* is asserted high.

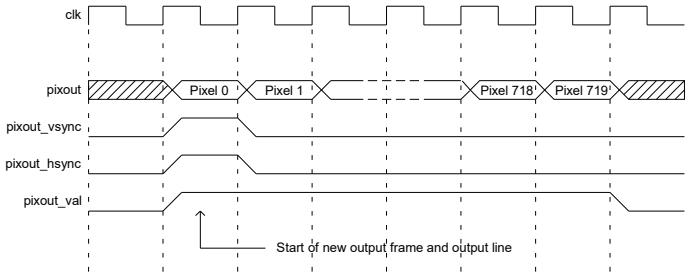

Figure 5: Output timing for the first line of a new frame

Source File Description

All source files are provided as text files coded in VHDL. The following table gives a brief description of each file.

Source file	***Description***
video_in.txt	Text-based source video file
deint_file_reader.vhd	Reads text-based source video file
deint_buffer.vhd	Input line buffer
deint_buffer_even.vhd	Input line buffer (one field only)
deint_filter_bob.vhd	Interpolation filter BOB
deint_filter_ela.vhd	Interpolation filter ELA
deint_filter_lci.vhd	Interpolation filter LCI
ram_dp_w_r.vhd	Dual port RAM component
deinterlacer.vhd	Top-level component
deinterlacer_bench.vhd	Top-level test bench

Functional Testing

An example VHDL testbench is provided for use in a suitable VHDL simulator. The compilation order of the source code is as follows:

1. deint_file_reader.vhd
2. deint_buffer.vhd
3. deint_buffer_even.vhd
4. deint_filter_bob.vhd
5. deint_filter_ela.vhd
6. deint_filter_lci.vhd
7. ram_dp_r_w.vhd
8. deinterlacer.vhd
9. deinterlacer_bench.vhd

The VHDL testbench instantiates the deinterlacer component and the user may modify the generic parameters in accordance with the desired interlaced video format and the desired filter algorithm. In the example provided, the input format has been set to 480i and the algorithm set to 'LCI'.

The source video for the simulation is read by the 'deint_file_reader' component. This component reads a text-based file which contains the RGB pixel data and sync information. The text file is called *video_in.txt* and should be placed in the top-level simulation directory.

The file *video_in.txt* follows a simple format which defines the state of signals: *pixin_val*, *pixin_field*, *pixin_vsync*, *pixin_hsync* and *pixin* on a clock-by-clock basis. An example file might be the following:

```
1  0  1  1  00 11 22  # pixel 0, line 0, start of  field 0
1  0  0  0  33 44 55  # pixel 1
1  0  0  0  66 77 88  # pixel 2
.
.
1  0  0  1  00 11 22  # pixel 0, line 1, field 0
1  0  0  0  33 44 55  # pixel 1
1  0  0  0  66 77 88  # pixel 2
.
.
1  1  1  1  00 11 22  # pixel 0, line 0, start of  field 1
1  1  0  0  33 44 55  # pixel 1
1  1  0  0  66 77 88  # pixel 2
.
.
1  1  0  1  00 11 22  # pixel 0, line 1, field 1
1  1  0  0  33 44 55  # pixel 1
1  1  0  0  66 77 88  # pixel 2
.
.
etc..
```

In this example, the first line of of the *video_in.txt* file asserts the input signals pixin_val = 1, pixin_field = 0, pixin_vsync = 1, pixin_hsync = 1 and pixin = 0x001122.

The simulation must be run for at least 10 ms during which time an output text file called *video_out.txt* will be generated. This file contains a sequential list of 24-bit output pixels. Figure 6 shows the resulting output frame generated by the test.

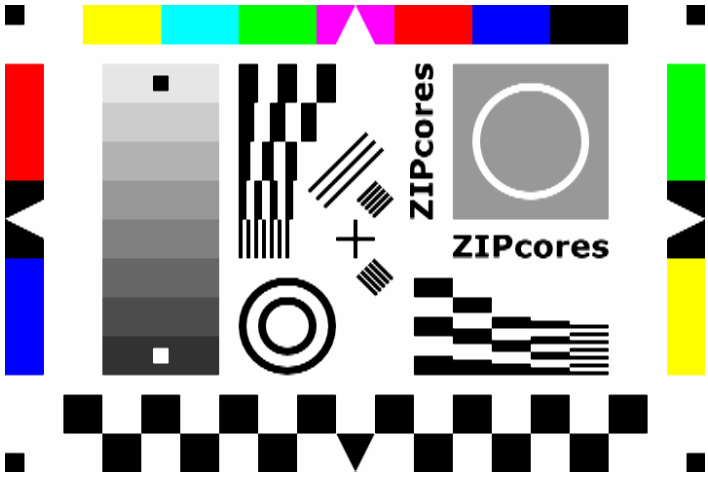

Figure 6: Output frame from testbench example

Performance

The deinterlacer core was tested using a varied selection of source images to enable the Peak Signal-to-Noise Ratio (PSNR) to be measured under different scenarios. Each source image was 720x576 pixels in resolution. The even lines were sampled from each source image to emulate a single field of a standard PAL 576i video signal.

The source video was passed through the deinterlacer hardware and the PSNR was calculated using the original source image as a reference. The PSNR measurements in dBs for each test image are shown in the table below.

PSNR (dB) for various test images

Image	*ID*	*BOB*	*ELA*	*LCI*	*Best*
Angelina	(a)	38.8	36.5	38.2	BOB
Blackbird	(b)	30.5	29.5	30.7	LCI
Chumps	(c)	44.3	43.7	43.6	BOB
Circuit	(d)	26.9	28.2	28.7	LCI
Fruit	(e)	31.5	29.7	31.4	BOB
Grass	(f)	33.7	33.6	34.7	LCI
Keyboard	(g)	30.4	30.3	30.6	LCI
Leaves	(h)	33.9	33.1	34.1	LCI
Lines	(i)	26.4	30.7	28.6	ELA
Spokes	(j)	27.0	26.4	27.3	LCI
Text	(k)	25.7	25.6	26.0	LCI
Watch	(l)	29.2	29.6	29.9	LCI
Average		31.5	31.4	32.0	LCI

Figure 7 shows the original source images used during the tests. The LCI algorithm was found to perform best overall. The BOB and ELA algorithms had similar average results.

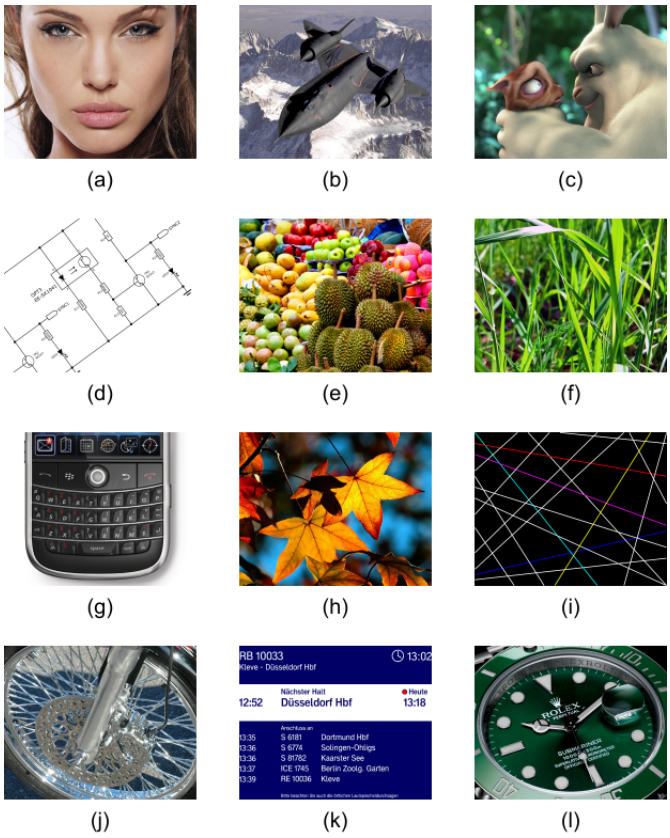

Figure 7: Source images used for PSNR measurements

Development Board Testing

The deinterlacer core was fully tested using a live PAL (576i) video source to review the subjective image quality for each of the filter algorithms. The basic setup included a Sony 'Handycam', a video decoder IC, a Spartan-6 FPGA to implement the deinterlacer IP Core and a video DAC connected to a flat-panel display. Figure 8 shows a simplified block diagram of the development system.

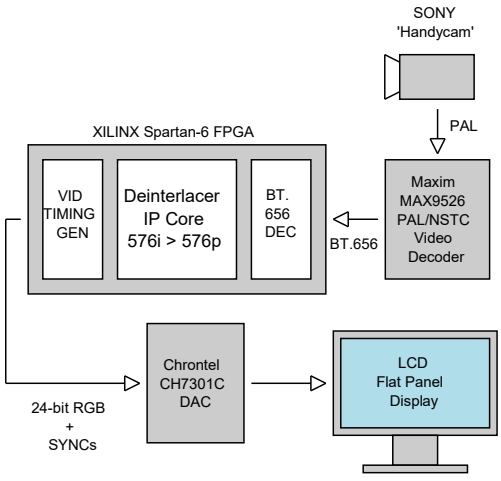

Figure 8: Development system hardware setup

Figure 9 is a photo of the basic hardware arrangement with the camera focussed on the edge of the oscilloscope. Different live video streams were used to review the subjective image quality.

Figure 9: Photo of the deinterlacer bench setup

It was found that BOB gave a good all round performance for natural video sequences. It did tend to soften the image more than ELA or LCI. For static images, BOB showed a minor vibration or perturbation between adjacent lines.

ELA gave a clean sharp image, but it did tend to increase the image contrast somewhat. Minor pixel displacements (especially around curved surfaces) were sometimes observed under close examination.

The LCI algorithm was found to give the most visually pleasing result for the widest range of video sources. The output video exhibited clean edges and excellent all round performance.

Synthesis

The files required for synthesis and the design hierarchy is shown below:

- deinterlacer.vhd
  - deint_buffer.vhd
  - deint_buffer_even.vhd
  - deint_filter_bob.vhd
  - deint_filter_ela.vhd
  - deint_filter_lci.vhd

The IP Core is designed to be technology independent. However, as a benchmark, synthesis results have been provided for the Xilinx® 7-series FPGAs. Synthesis results for other FPGAs and technologies can be provided on request.

Choosing the BOB algorithm results in the smallest and fastest implementation. The LCI algorithm results in a design roughly double in size. Careful attention must be made to the width of the line stores as this will effect the amount of RAM resource used.

Trial synthesis results are shown with the generic parameters set for PAL (576i) interlaced video using the LCI filter algorithm. The parameters were set as follows: *deint_mode* = 2, *frame_rate* = 1, *line_width* = 1024, *log2_line_width* = 10, *pixels_per_line* = 720, *lines_per_field* = 288, *field_polarity* = 0.

Resource usage is specified after place and route of the design.

XILINX® 7-SERIES FPGAS

Resource type	***Artix-7***	***Kintex-7***	***Virtex-7***
Slice Register	421	421	421
Slice LUTs	482	551	555
Block RAM	2	2	2
DSP48	0	0	0
Occupied Slices	220	239	246
Clock freq. (approx)	200 MHz	250 MHz	300 MHz

Revision History

Revision	***Change description***	***Date***
1.0	Initial revision	19/07/2010
1.1	Added PSNR performance data	03/08/2010
1.2	Clarified explanations for the different filter modes. Updated synthesis results in line with minor source code changes	08/03/2011
1.3	Made interlaced image dimensions fully programmable. Updated synthesis results for Xilinx® 7-series FPGAs	08/06/2016

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Development Software ***category:***

Click to view products by Zipcores ***manufacturer:***

Other Similar products are found below :

SRP004001-01 SW163052 SYSWINEV21 WS01NCTF1E W128E13 SW89CN0-ZCC IP-UART-16550 MPROG-PRO535E AFLCF-08-LX-CE060-R21 WS02-CFSC1-EV3-UP SYSMAC-STUDIO-EIPCPLR 1120270005 SW006021-2H ATATMELSTUDIO 2400573 2702579 2988609 SW006022-DGL 2400303 88970111 DG-ACC-NET-CD 55195101-101 55195101-102 SW1A-W1C MDK-ARM SW006021-2NH B10443 SW006021-1H SW006021-2 SW006022-2 SW006023-2 SW007023 MIKROE-730 MIKROE-2401 MIKROE-499 MIKROE-722 MIKROE-724 MIKROE-726 MIKROE-728 MIKROE-732 MIKROE-734 MIKROE-736 MIKROE-738 MIKROE-744 MIKROE-928 MIKROE-936 1120270002 1120270003 1120275015 NT-ZJCAT1-EV4