Logic - Five_Input_NOR, Interrupt, and LatchNoSeq

Example Code

```
| megaAVR Configurable Custom Logic library
| Five_input_NOR.ino
A library for interfacing with the megaAVR Configurable Custom Logic.
Developed in 2019 by MCUdude.
https://github.com/MCUdude/
In this example we use two logic blocks to get five inputs.
The output of block 1 is connected to one of the inputs of block 0.
| With the correct truth tables values we can make the output of
| block 0 go high when all inputs are low.
See Microchip's application note TB3218 for more information.
#include <Logic.h>
void setup() {
 // Initialize logic block 1
 // Logic block 1 has three inputs, PC0, PC1 and PC2 on ATmega parts
 // Logic block 1 inputs are PC3, PC4, and PC5 on ATtiny parts, but
 // only 24-pin parts have all three. 20-pin parts have PC3 only.
 // Because PA0 is not available on ATtiny under most situations
 // on megaTinyCore, we use input0 of logic block 0 as link from
 // other logic block. On the 20-pin parts, we also use the event system to
 // get the other two inputs from other pins.
 // Here, input2 is taken from the other
 // This example shows how this can be worked around for 20-pin parts
 // It's output is disabled because we connect the output signal to block 0
internally
 Logic1.enable = true;
                                   // Enable logic block 1
 Logic1.input0 = in::input_pullup; // Set PC0 as input with pullup
 Logic1.input1 = in::input_pullup; // Set PC1 as input with pullup
 Logic1.input2 = in::input_pullup; // Set PC2 as input with pullup
 Logic1.output = out::disable; // Enable output pin
 Logic1.filter = filter::disable; // No output filter enabled
 Logic1.truth = 0x01;
                                   // Set truth table
 // Initialize logic block 0
 // Logic block 0 has three inputs, PAO, PA1 and PA2.
 // Block 0 output on PA3 on ATmega, PA5 on ATtiny.
```

```
Logic0.enable = true;
                                     // Enable logic block 0
  Logic0.input0 = in::link;
                                     // Route output from block 1 to this input
internally
 Logic0.input1 = in::input_pullup; // Set PA1 as input with pullup
  Logic0.input2 = in::input_pullup; // Set PA2 as input with pullup
  Logic0.output = out::enable;
                                  // Enable logic block 0 output pin (PA3
(ATmega) or PA5 (ATtiny))
  LogicO.filter = filter::disable; // No output filter enabled
                                     // Set truth table
 Logic0.truth = 0xFE;
 // Initialize logic block 0 and 1
 Logic0.init();
 Logic1.init();
 // Start the AVR logic hardware
 Logic::start();
}
void loop() {
  // When using configurable custom logic the CPU isn't doing anything!
}
```

```
| megaAVR Configurable Custom Logic library
| Interrupt.ino
A library for interfacing with the megaAVR Configurable Custom Logic.
Developed in 2019 by MCUdude.
https://github.com/MCUdude/
| In this example we use the configurable logic peripherals the the
| megaAVR to create a 3-input NOR gate using logic block 2 on PORT D.
| We will use input on PD0, PD1 and PD2. Instead of having an output
pin the logic block will instead trigger an interrupt that runs a
user defined function.
                                3-input NOR truth table:
If we look at the truth table
                                |PD2|PD1|PD0| Y |
to the right, we can see that
                               |---|---|---|
all binary values for Y can
                                | 0 | 0 | 0 | 1 |
be represented as 00000001.
                                | 0 | 0 | 1 | 0 |
If we convert this 8-bit
                                | 0 | 1 | 0 | 0 |
| binary number into hex, we
                                0 1 1 1 0 1
get 0x01.
                                | 1 | 0 | 0 | 0 |
                                1 | 0 | 1 | 0 |
In this example the output is
                                | 1 | 1 | 0 | 0 |
true if all inputs are low.
                                | 1 | 1 | 1 | 0 |
```

```
#include <Logic.h>
void setup() {
  // Modify the serial port to match your hardware
  Serial.begin(9600);
 // The interrupt is only available on ATmega parts.
 // This will error on ATtiny parts.
 // Initialize logic block 2
 // Logic block 2 has three inputs, PAO, PA1 and PA2.
  // It has one output, but this is disabled because we're using an interrupt
instead.
  Logic2.enable = true;
                                     // Enable logic block 2
  Logic2.input0 = in::input_pullup; // Set PD0 as input with pullup
  Logic2.input1 = in::input_pullup; // Set PD1 as input with pullup
  Logic2.input2 = in::input_pullup; // Set PD2 as input with pullup
                                    // Disable output on PA0 (we don't have to
  Logic2.output = out::disable;
though)
  Logic2.filter = filter::disable; // No output filter enabled
  Logic2.truth = 0x01;
                                     // Set truth table
 // Initialize logic block 2
  Logic2.init();
  // Set interrupt (supports RISING, FALLING and CHANGE)
  Logic2.attachInterrupt(interruptFunction, RISING);
 // Start the AVR logic hardware
 Logic::start();
void loop() {
  // When using configurable custom logic the CPU isn't doing anything!
void interruptFunction() {
  Serial.println("Output of logic block 2 went high!");
}
```

```
/**********************************
| megaAVR Configurable Custom Logic library
|
| LatchNoSeq.ino
|
| A library for interfacing with the megaAVR Configurable Custom Logic. |
| Developed in 2019 by MCUdude.
| https://github.com/MCUdude/
```

```
In this example we use the configurable logic peripherals in AVR
  | to act as a "latch" WITHOUT using both LUTs and the sequencer
  For the even-numbered logic block(s) we can simply use the feedback
  | input. Otherwise we need to use the event system.
                                           3-input
                                                      truth table:
  | We use CCL LUT event as our "feedback", | PA2|PA1|CCL| Y |
  PA1 is RESET and PA2 is SET, both
                                           |---|---|---|
  | active low
                                           | 0 | 0 | 0 | 0 |
  Connect a button between those and Gnd | 0 | 0 | 1 | 1 |
  Pressing button on PA2 will set output
                                          | 0 | 1 | 0 | 1 |
  | HIGH and pressing button on PA1 will set | 0 | 1 | 1 | 1 |
  output LOW, and pressing neither will do | 1 | 0 | 0 | 0 |
  nothing.
                                           | 1 | 0 | 1 | 0 |
  | We could even then fire an interrupt from | 1 | 1 | 0 | 0 |
  | that pin
                                           The sky (well, and the number of LUTs) is the limit!!
  #include <Logic.h>
void setup() {
 // Initialize logic block 0
 // Logic block 0 has three inputs, PAO, PA1 and PA2.
 // Because PAO is shared with the UPDI pin on tinyAVR parts
 // we use the other two as our button inputs.
 // It outputs on the LUTO OUT pin - PA4 (alt. PB4) on ATtiny
 // or PA3 (alt PA6) everywhere else.
 Logic0.enable = true;
                                  // Enable logic block 0
 Logic0.input0 = in::feedback;
 Logic0.input1 = in::input_pullup;
                                                // PA1 as input1 (RESET)
 Logic0.input2 = in::input_pullup;
                                                // PA2 as input2 (SET)
 //Logic0.output_swap = out::pin_swap; // Uncomment this line to route the output
to alternate location, if available.
 Logic0.output = out::enable;
                                  // Enable logic block 0 output pin (see
pinout chart)
 LogicO.filter = filter::disable; // No output filter enabled
                                   // Set truth table - HIGH only if both high
 Logic0.truth = 0x8E;
 // Initialize logic block 0
 Logic0.init();
 // Example for odd-number block where we can't use feedback to get it's own
output
 #ifdef EVSYS CHANNELO //means it's not a 0/1-series
 EVSYS.CHANNEL0 = EVSYS_CHANNEL0_CCL_LUT1_gc;
 EVSYS.USERCCLLUT1A = EVSYS_USER_CHANNEL0_gc;
 #else //it's a tinyAVR 0/1
 EVSYS.ASYNCCH0 = EVSYS_ASYNCCH0_CCL_LUT1_gc; // Use CCL LUT1 as event
generator
 EVSYS.ASYNCUSER4 = EVSYS ASYNCUSER2 ASYNCCH0 gc; // ASYNCUSER4 is LUT1 event 0
```

```
#endif
  Logic1.input0 = in::event_a; // same distribution of inputs (though there
isn't the tinyAVR PA0 issue forcing it here)
  Logic1.input1 = in::input_pullup; // get it from LUT 1 event a or event 0
(tinyAVR 0/1 documentation calls them 0 and 1 - Logic.h accepts both for all
parts)
  Logic1.input2 = in::input pullup;
  //Logic0.output_swap = out::pin_swap; // Uncomment this line to route the output
to alternate location, if available.
  Logic1.output = out::enable;
                                     // Enable logic block 1 output pin (see
pinout chart)
  Logic1.filter = filter::disable; // No output filter enabled
                                     // Set truth table - HIGH only if both high
  Logic1.truth = 0x8E;
 // Initialize logic block 0
  Logic0.init();
  // Start the AVR logic hardware
 Logic::start();
}
void loop() {
 // When using configurable custom logic the CPU isn't doing anything!
}
```

Result

Examples compiled and uploaded successfully to the board.

Messages

C P B R R S P T S C S B P	NR Part hip Erase del AGEL S2 ESET disposit ETRY pulse erial programarallel programarallel programatabDelay mdexeDelay yncLoops yteDelay ollIndex ollValue emory Detail	tion n mode	2		: 0 us : P00 : P00	cated					
Polled				Block	Poll			Page			
	Memory Type	Mode De	elay	Size	Indx	Paged	Size	Size	#Pages	MinW	MaxW
ReadBack											
0x00 0x00	signature	0	0	0	0	no	3	0	0	0	0
	prodsig	0	0	0	0	no	125	125	0	0	0
0x00 0x00	fuses	0	0	0	0	no	9	16	0	0	0
0x00 0x00	C	0	0	0	0		1	0	0	0	0
0x00 0x00	fuse0	0	0	0	0	no	1	0	0	0	0
0x00 0x00	fuse1	0	0	0	0	no	1	0	0	0	0
0000 0000	fuse2	0	0	0	0	no	1	0	0	0	0
0x00 0x00	fuse4	0	0	0	a	no	1	0	0	0	0
0x00 0x00		O	O				_				
0x00 0x00	fuse5	0	0	0	0	no	1	0	0	0	0
CAGG CAGG	fuse6	0	0	0	0	no	1	0	0	0	0
0x00 0x00	fuse7	0	0	0	9	no	1	0	0	0	0
0x00 0x00											
0x00 0x00	fuse8	0	0	0	0	no	1	0	0	0	0
	lock	0	0	0	0	no	4	1	0	0	0
0x00 0x00	data	0	0	0	0	no	0	0	0	0	0
0x00 0x00		0	^				124070		_	_	
0x00 0x00	flash	0	0	0	0	no	131072	512	0	0	0
000 0 00	eeprom	0	0	0	0	no	512	32	0	0	0
0x00 0x00											

```
Programmer Type : JTAGICE3_UPDI
      Description
                : Microchip Curiosity in UPDI mode
      ICE hardware version: 0
      ICE firmware version: 1.17 (rel. 514)
      Serial number : MCHP3280031800001901
      Vtarget
                  : 3.31 V
      JTAG clock megaAVR/program: 0 kHz
      JTAG clock megaAVR/debug:
       JTAG clock Xmega: 0 kHz
      PDI clock Xmega : 100 kHz
avrdude: Partial Family_ID returned: "
avrdude: AVR device initialized and ready to accept instructions
avrdude: Device signature = 0x1e9708 (probably avr128da48)
avrdude: NOTE: "flash" memory has been specified, an erase cycle will be performed
      To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "0b11001001"
avrdude: writing fuse5 (1 bytes):
avrdude: 1 bytes of fuse5 written
avrdude: verifying fuse5 memory against 0b11001001:
avrdude: load data fuse5 data from input file 0b11001001:
avrdude: input file 0b11001001 contains 1 bytes
avrdude: reading on-chip fuse5 data:
avrdude: verifying ...
avrdude: 1 bytes of fuse5 verified
avrdude: reading input file "0x00"
avrdude: writing fuse7 (1 bytes):
avrdude: 1 bytes of fuse7 written
avrdude: verifying fuse7 memory against 0x00:
avrdude: load data fuse7 data from input file 0x00:
avrdude: input file 0x00 contains 1 bytes
avrdude: reading on-chip fuse7 data:
avrdude: verifying ...
avrdude: 1 bytes of fuse7 verified
avrdude: reading input file "0x00"
avrdude: writing fuse8 (1 bytes):
```

```
avrdude: 1 bytes of fuse8 written
avrdude: verifying fuse8 memory against 0x00:
avrdude: load data fuse8 data from input file 0x00:
avrdude: input file 0x00 contains 1 bytes
avrdude: reading on-chip fuse8 data:
avrdude: verifying ...
avrdude: 1 bytes of fuse8 verified
avrdude: reading input file
"C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/Five_input_NOR.ino.hex"
avrdude: writing flash (1636 bytes):
avrdude: 1636 bytes of flash written
avrdude: verifying flash memory against
C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/Five_input_NOR.ino.hex:
avrdude: load data flash data from input file
C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/Five_input_NOR.ino.hex:
avrdude: input file
C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/Five_input_NOR.hex
contains 1636 bytes
avrdude: reading on-chip flash data:
avrdude: verifying ...
avrdude: 1636 bytes of flash verified
avrdude done. Thank you.
```

Sketch uses 3224 bytes (2%) of program storage space. Maximum is 131072 bytes. Global variables use 565 bytes (3%) of dynamic memory, leaving 15819 bytes for local variables. Maximum is 16384 bytes.

avrdude: Version 6.3-20201216

Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/

Copyright (c) 2007-2014 Joerg Wunsch

System wide configuration file is

 $\label{thm:c:start} $$ "C:\Users\simeq \AppData\Local\Arduino15\packages\Microchip\hardware\megaav $$ r\1.0.0/avrdude.conf" $$$

Using Port : usb

Using Programmer : curiosity_updi

avrdude: Found CMSIS-DAP compliant device, using EDBG protocol

AVR Part : AVR128DA48

Chip Erase delay : 0 us PAGEL : P00 BS2 : P00

RESET disposition : dedicated

RETRY pulse : SCK serial program mode : yes parallel program mode : yes Timeout : 0 StabDelay : 0 CmdexeDelay : 0 SyncLoops : 0 ByteDelay : 0 PollIndex PollValue : 0x00

Memory Detail :

Polled					DIOCK	1011			rage			
	Memory	Туре	Mode	Delay	Size	Indx	Paged	Size	Size	#Pages	MinW	MaxW
ReadBack												
	signatu	ire	0	0	0	0	no	3	0	0	0	0
0x00 0x00	prodsig	ī	0	0	0	a	no	125	125	0	0	0
0x00 0x00	pi odsię	•	O	O	O	O	110	123	123	O	O	O
0,000 0,000	fuses		0	0	0	0	no	9	16	0	0	0
0x00 0x00	fuse0		0	0	0	0	no	1	0	0	0	0
0x00 0x00												
0x00 0x00	fuse1		0	0	0	0	no	1	0	0	0	0
0,000 0,000	fuse2		0	0	0	0	no	1	0	0	0	0
0x00 0x00	fuse4		0	0	0	0		1	0	0	0	0
0x00 0x00	Tuse4		0	0	0	0	no	1	0	0	0	0
	fuse5		0	0	0	0	no	1	0	0	0	0
0x00 0x00	fuse6		0	0	0	a	no	1	0	0	0	0
0x00 0x00	14500		Ü	Ü	Ü		110	-	Ü	Ü	Ü	Ü
000 000	fuse7		0	0	0	0	no	1	0	0	0	0
0x00 0x00	fuse8		0	0	0	0	no	1	0	0	0	0
0x00 0x00												
	lock		0	0	0	0	no	4	1	0	0	0

Block Poll

Page

	0 data	0	0	0	0 no	a	0	0	0	0
0x00 0x0		0	O	O	0 110	0	0	0	0	O
0,000 0,00	flash	0	0	0	0 no	131072	512	0	0	0
0x00 0x0	0									
	eeprom	0	0	0	0 no	512	32	0	0	0
0x00 0x0	0									
	Description ICE hardware ICE firmware Serial numbe Vtarget JTAG clock m JTAG clock m JTAG clock X PDI clock Xm	e version: e version: er : MCH : 3.3 negaAVR/pr negaAVR/de	0 1.17 IP32800 1 V cogram:	(rel. 031800 : 0 kH	514) 001901 z	UPDI mode	•			
	Partial Fami AVR device i	.ly_ID ret	urned:		" to acce	ot instruc	tions			
Reading	##########	+#########	#####	#####	#######	#########	# 100	% 0.019	5	
	Device signa NOTE: "flash	" memory	has be	een sp	ecified,	an erase	cycle w	ill be	perfor	rmed
avrdude:	To disable terasing chip reading input	o ut file "0	b11001		CHE -D	option.				
avrdude: avrdude:	erasing chip reading inpu	o ut file "0 25 (1 byte	b11001 s):	1001"			:# 100	% 0.02s	5	
avrdude: avrdude: Writing	erasing chip reading inpu writing fuse	o ut file "0 25 (1 byte	0b11001 es):	1001"			# 100	% 0.02s	5	
avrdude: avrdude: Writing avrdude:	erasing chip reading inpu writing fuse ####################################	nt file "0 e5 (1 byte ####################################	0b11001 es): ######	1001" #####	#######		# 100	% 0.029	5	
avrdude: avrdude: Writing avrdude: avrdude:	erasing chip reading inpu writing fuse ########## 1 bytes of f verifying fu	nt file "0 e5 (1 byte e###################################	b11001 es): ##### eten ey agai	1001" #####	####### b1100100	######################################	# 100	% 0.02s	5	
avrdude: avrdude: Writing avrdude: avrdude: avrdude:	erasing chip reading inpu writing fuse ####################################	et file "0 e5 (1 byte e######## Guse5 writ use5 memor use5 data	b11001 es): ###### eten ey agai	1001" ##### inst 0	####### b1100100 file 0b1	######################################	# 100	% 0.025	5	
avrdude: avrdude: Writing avrdude: avrdude: avrdude:	erasing chip reading inpu writing fuse ####################################	out file "0 e5 (1 byte f######## Fuse5 writ use5 memor use5 data ob11001001	ob11001 es): ##### eten ey agai from i	1001" ##### inst 0 input	####### b1100100 file 0b1	######################################	# 100	% 0.02s	5	
avrdude: avrdude: writing avrdude: avrdude: avrdude: avrdude: avrdude:	erasing chip reading inpu writing fuse ########## 1 bytes of f verifying fu load data fu input file 0	et file "0 e5 (1 byte ######### Fuse5 writ use5 memor use5 data ub11001001	bb11001 es): ###### ten y agai from i	inst 0 input ains 1	####### b1100100 file 0b1 bytes	########## 1: 1001001:				
avrdude:	erasing chip reading inpu writing fuse ######### 1 bytes of f verifying fu load data fu input file @ reading on-o	et file "0 es (1 byte es (1 byte es write es write es best data es data es fuses es data es fuses es data es fuses es data es data es data es fuses	b11001 es): ###### eten ey agai from i conta data: ######	inst 0 input ains 1	####### b1100100 file 0b1 bytes	########## 1: 1001001:				
avrdude:	erasing chip reading input writing fuse ########### 1 bytes of for verifying furified data furified data furified for eading on-comparing furified for eading input file for eading input file for eading input furifying 1 bytes of for reading input furified for eading input furified furified for eading input furified for eading input furified furified for eading input furified	t file "0 t file "0 t file "0 t ######## fuse5 writ use5 memor use5 data bb11001001 thip fuse5 t####################################	b11001 es): ###### eten ey agai from i conta data: #######	inst 0 input ains 1	####### b1100100 file 0b1 bytes #######	"######## 1: 1001001:	# 100	% 0.00s	5	

```
avrdude: verifying ...
avrdude: 1 bytes of fuse7 verified
avrdude: reading input file "0x00"
avrdude: writing fuse8 (1 bytes):
avrdude: 1 bytes of fuse8 written
avrdude: verifying fuse8 memory against 0x00:
avrdude: load data fuse8 data from input file 0x00:
avrdude: input file 0x00 contains 1 bytes
avrdude: reading on-chip fuse8 data:
avrdude: verifying ...
avrdude: 1 bytes of fuse8 verified
avrdude: reading input file
"C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/Interrupt.ino.hex"
avrdude: writing flash (3224 bytes):
avrdude: 3224 bytes of flash written
avrdude: verifying flash memory against
C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/Interrupt.ino.hex:
avrdude: load data flash data from input file
C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/Interrupt.ino.hex:
avrdude: input file
C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/Interrupt.ino.hex
contains 3224 bytes
avrdude: reading on-chip flash data:
avrdude: verifying ...
avrdude: 3224 bytes of flash verified
avrdude done. Thank you.
```

Sketch uses 1684 bytes (1%) of program storage space. Maximum is 131072 bytes. Global variables use 222 bytes (1%) of dynamic memory, leaving 16162 bytes for

local variables. Maximum is 16384 bytes.

avrdude: Version 6.3-20201216

Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/

Copyright (c) 2007-2014 Joerg Wunsch

System wide configuration file is

"C:\Users\ivanFernandez\AppData\Local\Arduino15\packages\Microchip\hardware\megaavr\1.0.0/avrdude.conf"

Using Port : usb

Using Programmer : curiosity_updi

avrdude: Found CMSIS-DAP compliant device, using EDBG protocol

AVR Part : AVR128DA48

Chip Erase delay : 0 us PAGEL : P00 BS2 : P00

RESET disposition : dedicated

RETRY pulse : SCK serial program mode : yes parallel program mode : yes Timeout : 0 StabDelay : 0 CmdexeDelay : 0 : 0 SyncLoops ByteDelay : 0 PollIndex : 0 PollValue : 0x00

Memory Detail :

Block Poll Page Polled Memory Type Mode Delay Size Indx Paged Size Size #Pages MinW MaxW ReadBack signature 0 0 0 no 3 0 0x00 0x00 125 125 0 0 0 prodsig 0 no 0 0 0x00 0x00 0 0 9 16 0 fuses 0 no 0x00 0x00 0 0 0 fuse0 0 no 1 0 0x00 0x00 0 no fuse1 0x00 0x00 0 0 0 fuse2 0 0 no 0 0 0 0x00 0x00 fuse4 0 0 0 no 1 0 0 0x00 0x00 fuse5 0 0 0 0 no 1 0 0 0 0 0x00 0x00 0 fuse6 1 0 no

0x00 0x0	00 fuse7	0	0	0	0 no	1	0	0	0	0	
0x00 0x0											
0x00 0x0	fuse8 00	0	0	0	0 no	1	0	0	0	0	
	lock	0	0	0	0 no	4	1	0	0	0	
0x00 0x0	00 data	0	0	0	0 no	0	0	0	0	0	
0x00 0x0		O	0	0	0 110	0	0	O	0	O	
0,400, 0,40	flash	0	0	0	0 no	131072	512	0	0	0	
0x00 0x0	eeprom	0	0	0	0 no	512	32	0	0	0	
0x00 0x0	00										

Programmer Type : JTAGICE3_UPDI

Description : Microchip Curiosity in UPDI mode

ICE hardware version: 0

ICE firmware version: 1.17 (rel. 514)
Serial number : MCHP3280031800001901

Vtarget : 3.31 V

JTAG clock megaAVR/program: 0 kHz
JTAG clock megaAVR/debug: 0 kHz

JTAG clock Xmega: 0 kHz PDI clock Xmega: 100 kHz

avrdude: Partial Family_ID returned: " "

avrdude: AVR device initialized and ready to accept instructions

avrdude: Device signature = 0x1e9708 (probably avr128da48)

avrdude: NOTE: "flash" memory has been specified, an erase cycle will be performed

To disable this feature, specify the -D option.

avrdude: erasing chip

avrdude: reading input file "0b11001001"

avrdude: writing fuse5 (1 bytes):

avrdude: 1 bytes of fuse5 written

avrdude: verifying fuse5 memory against 0b11001001:

avrdude: load data fuse5 data from input file 0b11001001:

avrdude: input file 0b11001001 contains 1 bytes

avrdude: reading on-chip fuse5 data:

avrdude: verifying ...

avrdude: 1 bytes of fuse5 verified avrdude: reading input file "0x00" avrdude: writing fuse7 (1 bytes):

```
avrdude: 1 bytes of fuse7 written
avrdude: verifying fuse7 memory against 0x00:
avrdude: load data fuse7 data from input file 0x00:
avrdude: input file 0x00 contains 1 bytes
avrdude: reading on-chip fuse7 data:
avrdude: verifying ...
avrdude: 1 bytes of fuse7 verified
avrdude: reading input file "0x00"
avrdude: writing fuse8 (1 bytes):
avrdude: 1 bytes of fuse8 written
avrdude: verifying fuse8 memory against 0x00:
avrdude: load data fuse8 data from input file 0x00:
avrdude: input file 0x00 contains 1 bytes
avrdude: reading on-chip fuse8 data:
avrdude: verifying ...
avrdude: 1 bytes of fuse8 verified
avrdude: reading input file
"C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/LatchNoSeq.ino.hex"
avrdude: writing flash (1684 bytes):
avrdude: 1684 bytes of flash written
avrdude: verifying flash memory against
C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/LatchNoSeq.ino.hex:
avrdude: load data flash data from input file
C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/LatchNoSeq.ino.hex:
avrdude: input file
C:\Users\IVANFE~1\AppData\Local\Temp\arduino_build_59380/LatchNoSeq.ino.hex
contains 1684 bytes
avrdude: reading on-chip flash data:
avrdude: verifying ...
avrdude: 1684 bytes of flash verified
avrdude done. Thank you.
```

Notes and Potential Fixes

1. Each of the sketches compiled and uploaded successfully to the AVR128DA48 board. This concludes testing of the Logic examples within the Team 25 core.