Now I'm confused, probably yours is right, but nothing is making sense...

colombod commented on 6 Jun 2013 Contributor

So ... what is the last word here?

pklaus commented on 7 Jun 2013 Contributor

OK, here is a quick clarification of the pin usage: The AT91SAM3X8E has 9 general-purpose 32-bit timers/counters; they have input lines for their clock and input/output lines that can be used in different ways, such as PWM outputs etc.

Here is a quick overview of the signals of the Timer/Counter part of the AT91SAM3X8E:

Instance	TC	Channel	External Clock Input	I/O Line A	I/O Line B
T0	TC0	0	TCLK0	TIOA0	TIOB0
T1	TC0	1	TCLK1	TIOA1	TIOB1
T2	TC0	2	TCLK2	TIOA2	TIOB2
T3	TC1	0	TCLK3	TIOA3	TIOB3
T4	TC1	1	TCLK4	TIOA4	TIOB4
T5	TC1	2	TCLK5	TIOA5	TIOB5
T6	TC2	0	TCLK6	TIOA6	TIOB6
T7	TC2	1	TCLK7	TIOA7	TIOB7
T8	TC2	2	TCLK8	TIOA8	TIOB8

Here is how the external clock inputs are routed to pins on the Arduino Due board:

Clock Input	Port Pin of µC	Pin on Arduino Due Board
TCLK0	PB 26	Digital Pin 22
TCLK1	PA 4	Analog In 5
TCLK2	PA 7	Digital Pin 31
TCLK3	PA 22	Analog In 3
TCLK4	PA 23	Analog In 2
TCLK5	PB 16	DAC1
TCLK6	PC 27	/
TCLK7	PC 30	LED "RX"
TCLK8	PD 9	Digital Pin 30

Here is how the *I/O Lines A* are routed to pins on the Arduino Due board:

I/O Line A	Port Pin of µC	Pin on Arduino Due Board
TIOA0	PB 25	Digital Pin 2
TIOA1	PA 2	Analog In 7
TIOA2	PA 5	/
TIOA3	PE 9	/
TIOA4	PE 11	/
TIOA5	PE 13	/
TIOA6	PC 25	Digital Pin 5
TIOA7	PC 28	Digital Pin 3
TIOA8	PD 7	Digital Pin 11

I/O Line A Port Pin of µC Pin on Arduino Due Board

Here is how the *I/O Lines B* are routed to pins on the Arduino Due board:

I/O Line B	Port Pin of µC	Pin on Arduino Due Board
TIOB0	PB 27	Digital Pin 13 / Amber LED "L"
TIOB1	PA 3	Analog In 6
TIOB2	PA 6	Analog In 4
TIOB3	PE 10	/
TIOB4	PE 12	/
TIOB5	PE 14	/
TIOB6	PC 26	Digital Pin 4 (also connected to PA29)
TIOB7	PC 29	Digital Pin 10 (also connected to PA28)
TIOB8	PD 8	Digital Pin 12

This library uses the **Timer Counter** as a timer to run a callback. It doesn't need the external clock inputs as it uses the built-in clock inputs for the TCs. As the Timer/Counters are being driven in the "Waveform Mode", this generates a PWM signal on the TIOAx / TIOBx lines (which we don't need if all we want from this code is to run a callback).

I don't know what happens to TIOAx and TIOBx if you set up the output controller to ignore all possible impact factors on the output state. This could possibly be done by replacing the line with the `TC_Configure()` call with

```
uint32_t disable_TIO = TC_CMR_ACPC_NONE | TC_CMR_ACPA_NONE |
    TC_CMR_AEEVT_NONE | TC_CMR_ASWTRG_NONE |
    TC_CMR_BCPC_NONE | TC_CMR_BCPB_NONE |
    TC_CMR_BEEVT_NONE | TC_CMR_BSWTRG_NONE;
TC_Configure(t.tc, t.channel, TC_CMR_WAVE | TC_CMR_WAVSEL_UP_RC | clock | disable_TIO);
```

You may or may not be able to use TIOAx and TIOBx as your own output in this case. **I didn't test this yet!**

My sources: Section *37.5.1 I/O Lines* in the Atmel SAM3X Datasheet, Arduino Due's SAM3X Pin Mapping and the Arduino Due pinout diagram.

pklaus referenced this issue on 8 Jun 2013

README transferred to Markdown format #15 Merged

pklaus commented on 8 Jun 2013 Contributor

I was reading the datasheet some more and testing the code and I come to the following findings:
You don't even need to set the flags above as I though, as we do not tell the PIO controller of the µC to connect the hardware pins to the corresponding peripherals. The pins are still free to be used in any way you want. So it doesn't matter to the user if the timer has its TIOAx or TIOBx lines mapped to pins or not if all they want is to use the timer to run their callback function.
Thus we can remove the sentence from the README.

If we ever want to use the timer / counter capabilities to route the TIOAx signal to a pin on the outside (for a PWM), we would have to set up the PIO like this:

```
int ulPin = 2; // just an example:   it's 2  for the Timer0
PIO_Configure(g_APinDescription[ulPin].pPort,
    g_APinDescription[ulPin].ulPinType,
    g_APinDescription[ulPin].ulPin,
    g_APinDescription[ulPin].ulPinConfiguration);
```

I found this fragment in the file wiring_analog.c of the Arduino 1.5.x branch.

ivanseidel closed this in `170a12d` on 8 Jun 2013

sriranjanr commented on 21 Jun 2015