Application Note

Busemulation

History

13th Oct. 99	MM	V1.0	New Format, new updated version
04th Jul. 00	MEN	V1.1	Updated Application

Warranty and Disclaimer

To the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH restricts its warranties and its liability for **all products delivered free of charge** (eg. software include or header files, application examples, application Notes, target boards, evaluation boards, engineering samples of IC's etc.), its performance and any consequential damages, on the use of the Product in accordance with (i) the terms of the License Agreement and the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials. In addition, to the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH disclaims all warranties and liabilities for the performance of the Product and any consequential damages in cases of unauthorised decompiling and/or reverse engineering and/or disassembling. **Note, all these products are intended and must only be used in an evaluation laboratory environment**.

1. Fujitsu Mikroelektronik GmbH warrants that the Product will perform substantially in accordance with the accompanying written materials for a period of 90 days form the date of receipt by the customer. Concerning the hardware components of the Product, Fujitsu Mikroelektronik GmbH warrants that the Product will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer.

2. Should a Product turn out to be defect, Fujitsu Mikroelektronik GmbH´s entire liability and the customer´s exclusive remedy shall be, at Fujitsu Mikroelektronik GmbH´s sole discretion, either return of the purchase price and the license fee, or replacement of the Product or parts thereof, if the Product is returned to Fujitsu Mikroelektronik GmbH in original packing and without further defects resulting from the customer´s use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to Fujitsu Mikroelektronik GmbH, or abuse or misapplication attributable to the customer or any other third party not relating to Fujitsu Mikroelektronik GmbH.

3. To the maximum extent permitted by applicable law Fujitsu Mikroelektronik GmbH disclaims all other warranties, whether expressed or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the Product is not designated.

4. To the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH´s and its suppliers´ liability is restricted to intention and gross negligence.

   NO LIABILITY FOR CONSEQUENTIAL DAMAGES

   To the maximum extent permitted by applicable law, in no event shall Fujitsu Mikroelektronik GmbH and its suppliers be liable for any damages whatsoever (including but without limitation, consequential and/or indirect damages for personal injury, assets of substantial value, loss of profits, interruption of business operation, loss of information, or any other monetary or pecuniary loss) arising from the use of the Product.

Should one of the above stipulations be or become invalid and/or unenforceable, the remaining stipulations shall stay in full effect.

In some applications, MCUs represent a perfect set of resources, but with the lack of a bus interface. If a main processor (or a high-performance controller) is present on this design, the solution can be to establish a software-based bus interface. This application note shows how to control an MB90F594 (16-LX device) from an MB91101 (FR30 based MCU). In this particular case, the purpose was to add some peripheral functions (esp. CAN) to the MB91101 externally. From the software perspective, the aim was to treat the "external" resources on the MB90F594 in extactly the same way than the internal resources.

Implementation

The bus-interface of the MB91101 is quite flexible due to six fully programmable chip-select outputs, additional control-lines and variable wait-states per CS-area. On the other side, the 16-LX device uses 5 (8-Bit) parallel ports to connect to the bus : A 12-Bit addressbus (for accessing the lower and upper half of the IO-Area), a 16-Bit Databus and 5 additional control-lines. In addition, two (8-Bit) latches and a programmable logic device (PLD) are needed to complete the interface. The software running on the MB90F594 only handles the bus interface, so that the 16-bit MCUs appears to be an "intelligent resource-container" where a set of control- and data-registers (IO-Area) is transparent from outside. Fig. 1 shows a block diagram of the two controllers and the interface :

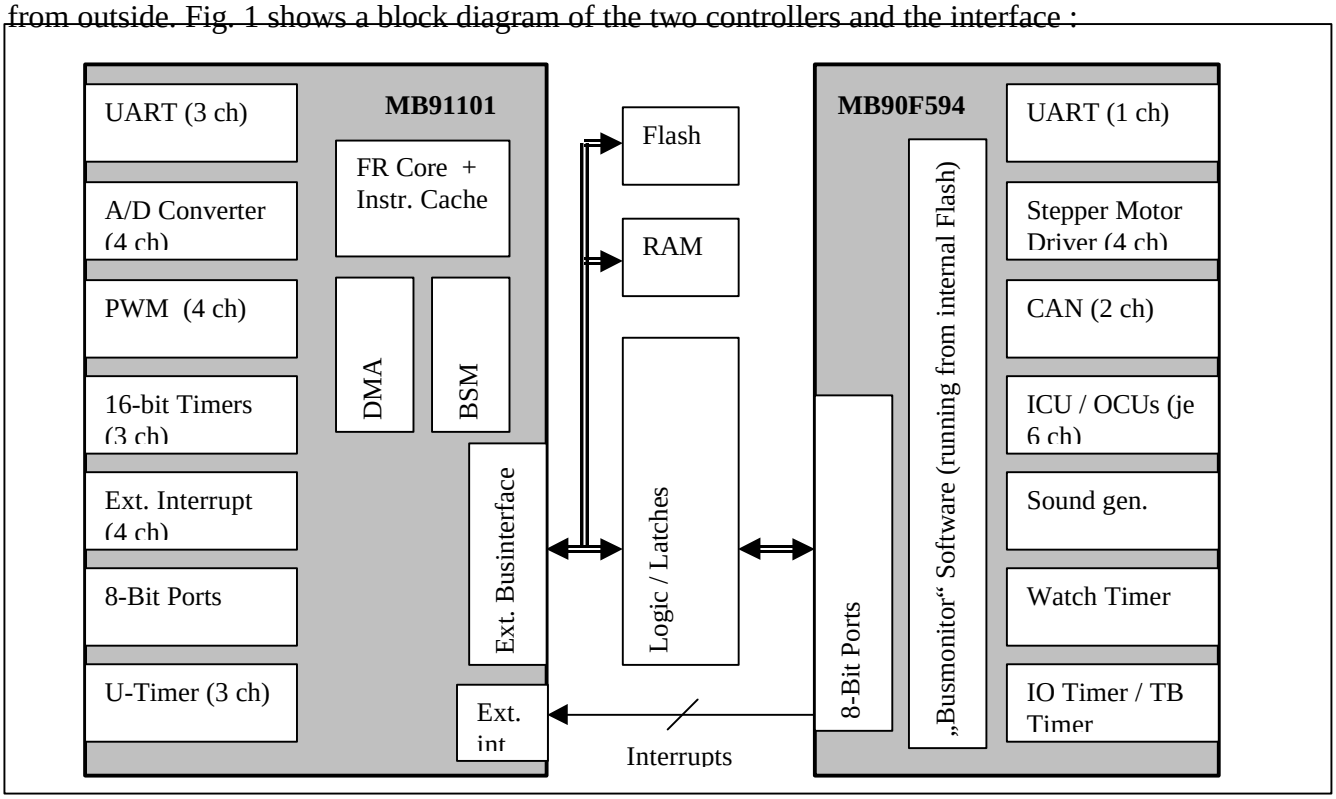

Fig 1 : Blockdiagram of the main controller (MB91101) and a secondary device (MB90F594) accessed via the ext. bus

Additional Businterface Hardware

If the PLD detects an access to the chip-select area, it pulls the READY-line low in order to compensate the difference in performance between the devices. Then, a read or write access will be signaled to the 16-Bit MCU as well as the data-width (8- or 16-bit). Futhermore, the latches will be set “transparent” in the appropriate direction for the duration of the access. This is important, because other devices hooked up to the bus must not be disturbed.
If the access has been completed, the READY-line will be released and the MB91101 continues execution. In order to maintain the given bus-timings, the PLD works in a clock-syncronous way (external bus-clock from MB91101 is connected to the PLD). Fig. 2 shows the interconnection of control- and data lines.

Fig. 3 shows the PLD state-diagram : After power-up, various outputs will be initialized, then the device waits for the read (lower path) or write (upper path) conditions. The CSDC line triggers the MB90F594 (the RD-line will also be read by the MB90F594). By monitoring the BUSY (Port 3.2) output from the MB90F594, the PLD will know exactly when the current access has been completed. For security reasons, the process continues after the release of the RD- oder WRx line. Fig. 4 and 5 show the logic simulation results.

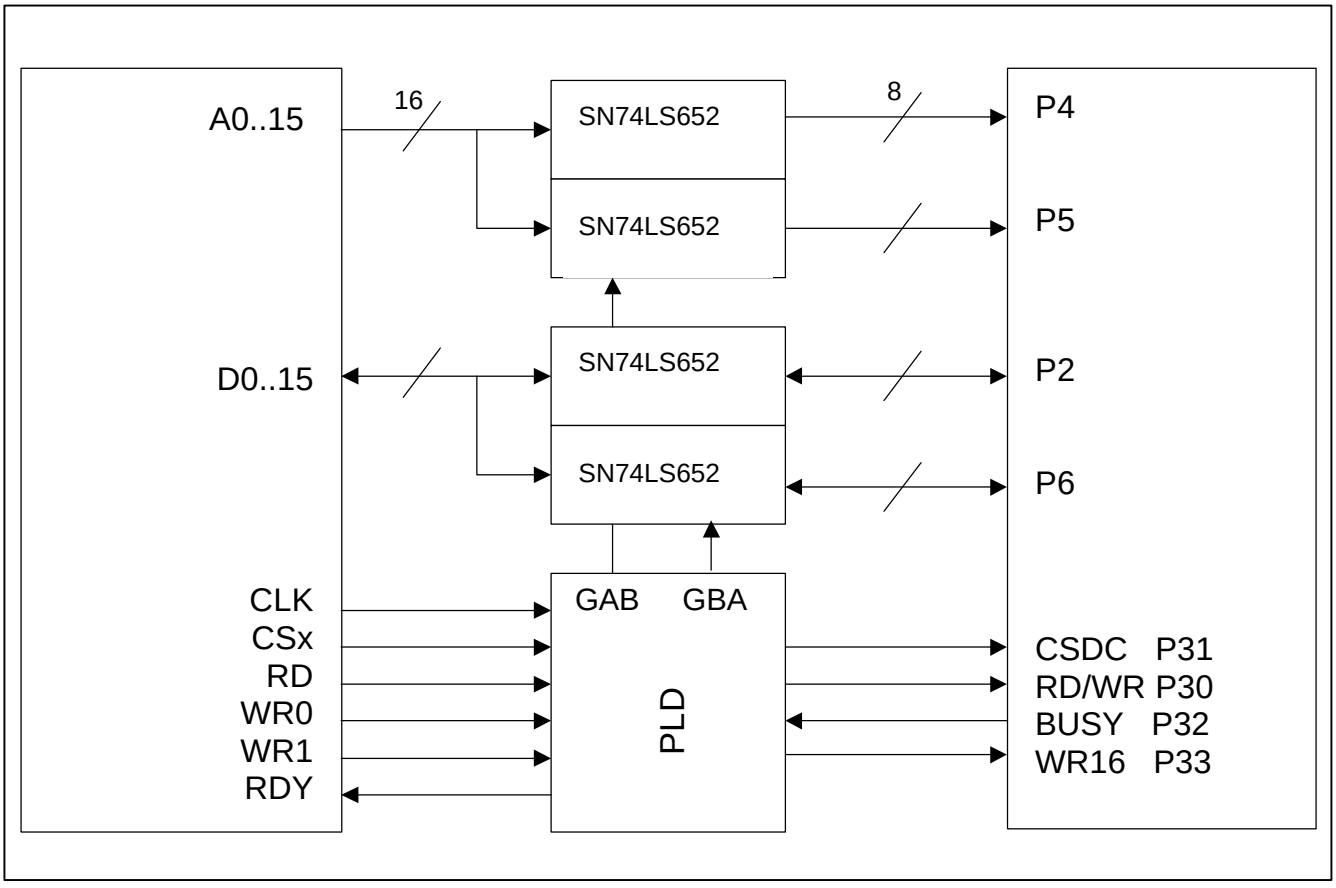

Fig. 2 : Interconnections of bus hardware

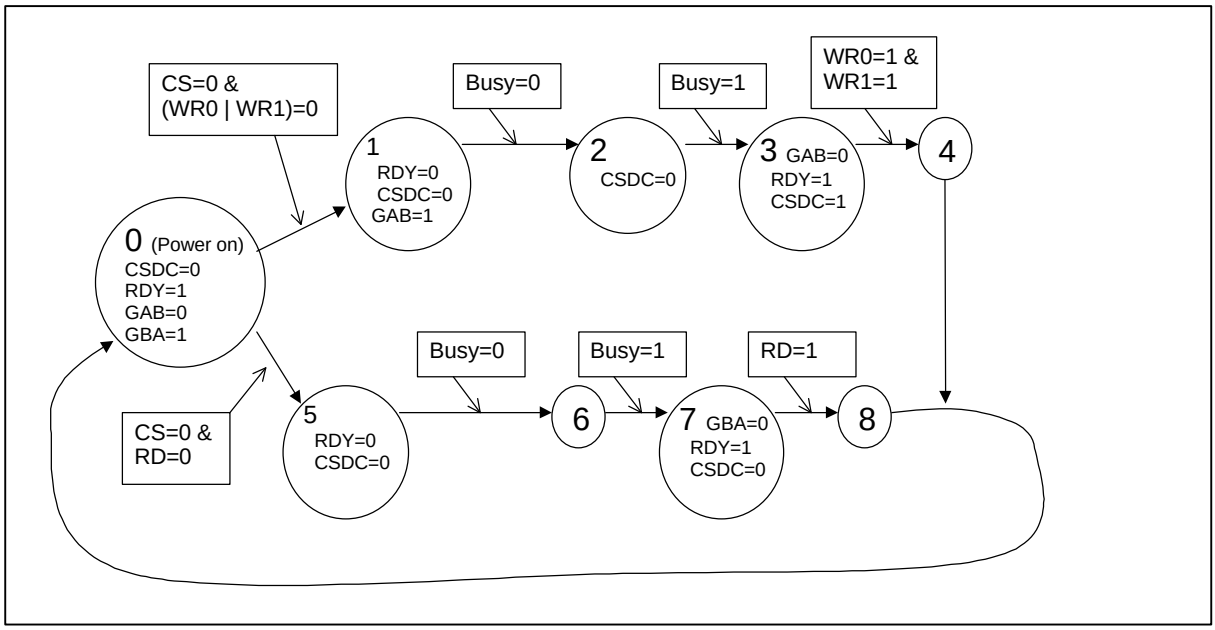

Fig. 3 : PAL „Statemachine“ overview

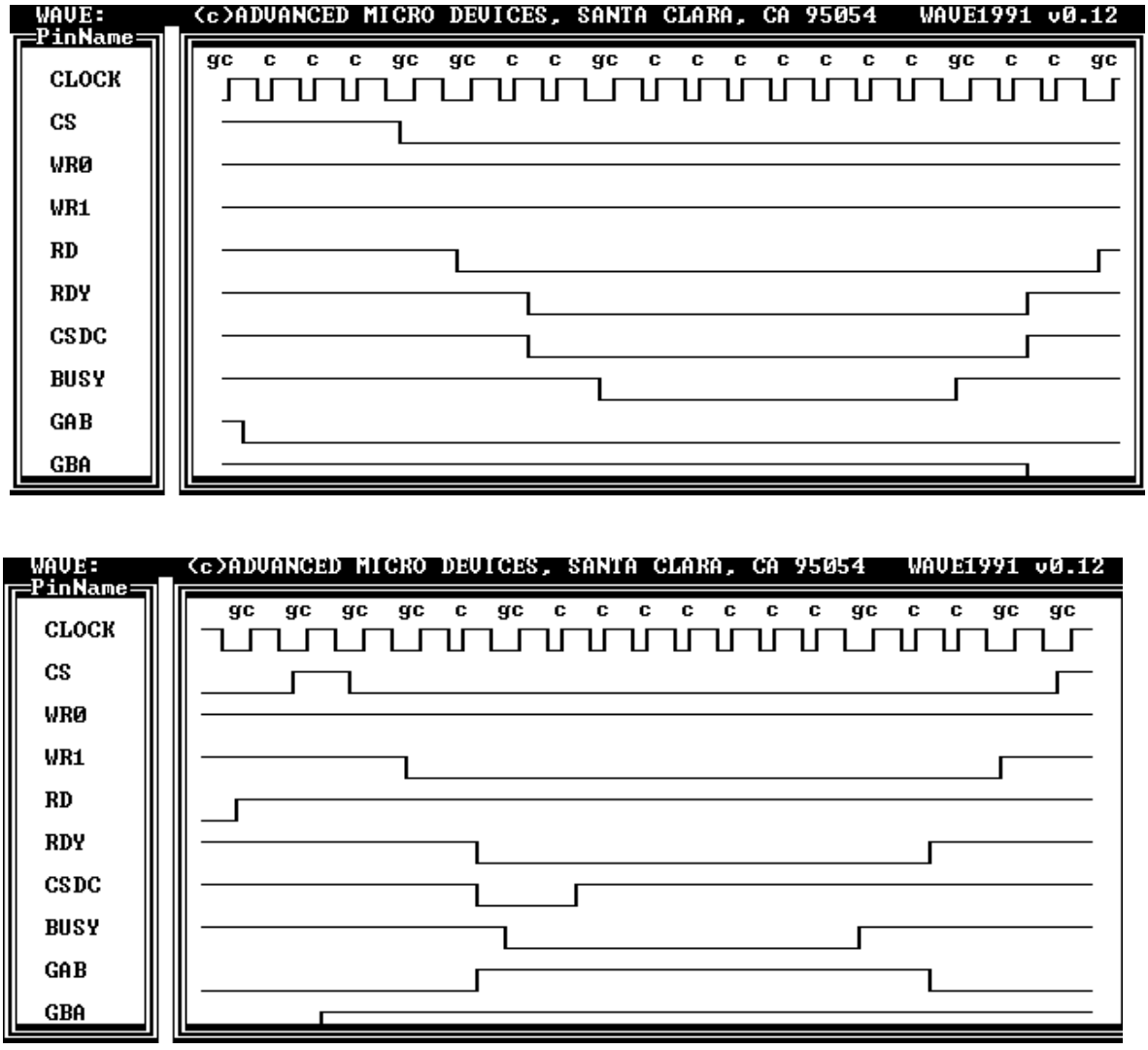

Fig. 4 : Buslines and typical behaviour (Simulation) of the PLD
upper diagram : Read access from the MB90F594
lower diagram : Write access to the MB90F594

Interrupts

One of the most important features of modern MCUs is the possibility for resources to signal important events to the CPU asyncronously - Interrupts. In this case, interrupts from an MB90F594's peripheral need also to be signaled to the FR30-core. The following interrupt-routing principle was used: The software on the MB90F594 reads special configuration registers which enables IRQ routing for various interrupts. These IRQ flags will be shown inverted on dedicated port-pins and these pins can be connected to the external interrupts of the MB91101. Using this technique, external resources on the MB90F594 can signal "real" interrupts to the FR30-core and the software has to treat the interrupt request similar to internal sources. Of course, the number is limited to the number of available external interrupts (in this case 4) and the interrupt latancy will be increased. But for low-frequency interrupts (e.g. CAN receive), this method is quite useful. Unlike using fixed-wired interrupts in standard MCUs, the 4 interrupt-sources in this system can be reconfigured to a different source anytime during execution.

Software

It was already mentioned that programming the additional resources on the MB90F594 is no different from programming one of the MB91101-internals. Assuming the following modules to be present, code can be created with no limitation :

1. The external chip-select area must be initialized correctly (should be included in `Startup.Asm`)
2. A special include file must be available which defines all control- and data registers externally (Note : Some of the IO-registers may have the same name as : e.g. PDR2, therefore a suitable prefix should be used, in this case, all identifiers starting with an "X" are LX-registers)
3. Interrupts (if used) must be enabled as external interrupts and configured using the appropriate routing registers

This example shows how to use the "Sound generator" within the MB90F594 (interrupt-controlled) :

```
/*****************************************************************************/
void main(void)
{
    initbus();                /* init bus for external MB90F594 */
    XINT0 = XINT0_SOUND;      /* Sound interrupt routed to ext. INT0 */
    ENIR = 1;                 /* allow INT0 */
    ICR00 = 20;               /* level for INT0 */
    vectors[255-16] = int0isr;  /* define interrupt vector for INT0 */
    __EI();                   /* enable interrupts */
    __set_il(30);             /* set Int-level */
    InitXSoundGen();          /* start sound */
    while(1);                 /* dummy loop (wait for int) */
}
/*****************************************************************************/
/* Interrupt Service Routine – an interrupt will be signaled after each tone */
/*****************************************************************************/
void __interrupt int0isr(void)
{
         EIRR_ER0 = 0;               /* clear int flags */
         XSGCR_INT = 0;
         XSGAR = 0xA0;               /* set amplidude to it's initial value */
         switch(XSGFR)
         {                           /* 3-tone "gong" sound */
           case 0x03: XSGFR = 0x05;  break;
           case 0x05: XSGFR = 0x08;  break;
           case 0x08: XSGFR = 0x03;  break;
         }
}
```

Markus Mierse