**Freescale Semiconductor**
Application Note

Document Number: AN3628
Rev. 0, 04/2008

# Creating an External Bus Interface Using Rapid GPIO and Timers

by: Martin Latal
Technical Information Center, Roznov

## 1 Introduction

This application note provides examples of configuring and using MCF51QE128 timer and Rapid GPIO modules. The flexibility of these two modules is demonstrated by implementation of a general-purpose parallel bus interface.

## 2 Rapid GPIO

This section demonstrates the advantages of the Rapid GPIO module over a parallel input/output control module.

### 2.1 General Overview

The Rapid GPIO module provides a 16-bit general purpose I/O module directly connected to the processor's high-speed 32-bit local platform bus. This connection supports single-cycle, zero-wait-state data transfers, therefore providing improved pin performance over

**Contents**





freescale semiconductor

traditional GPIO modules located on the internal slave peripheral bus.

## 2.2 Rapid GPIO Application Example

This application example shows the Rapid GPIO set/clear switching frequency. In this example, the V1 core is running at a frequency of 50 MHz. This code snippet will explain the test:

```
/* settings for RGPIO module registers */
RGPIO_ENB = 0xFFFF;//enable RGPIO pins
RGPIO_DATA = 0x00;//zeroes
RGPIO_DIR = 0xFFFF;//outputs

/* RGPIO move test */
asm
{
     //Prepare D4 and D5 for future use
     clr.l   d4
     clr.l   d5
     move.w   #0xFFFF, d5

     move.w  d5, RGPIO_DATA//will drive the pins high
     move.w  d4, RGPIO_DATA//will drive the pins low
     move.w  d5, RGPIO_DATA
     move.w  d4, RGPIO_DATA
     ...
}
```

**Example 1. RGPIO Set/Clear Square Wave Generation**

First, D4 and D5 registers are initialized. The core then executes store instructions (moves the contents of the internal registers to the RGPIO_DATA register), which will effectively toggle the RGPIO output pins. The signal can be captured with a scope. The resulting signal is shown in Figure 1.

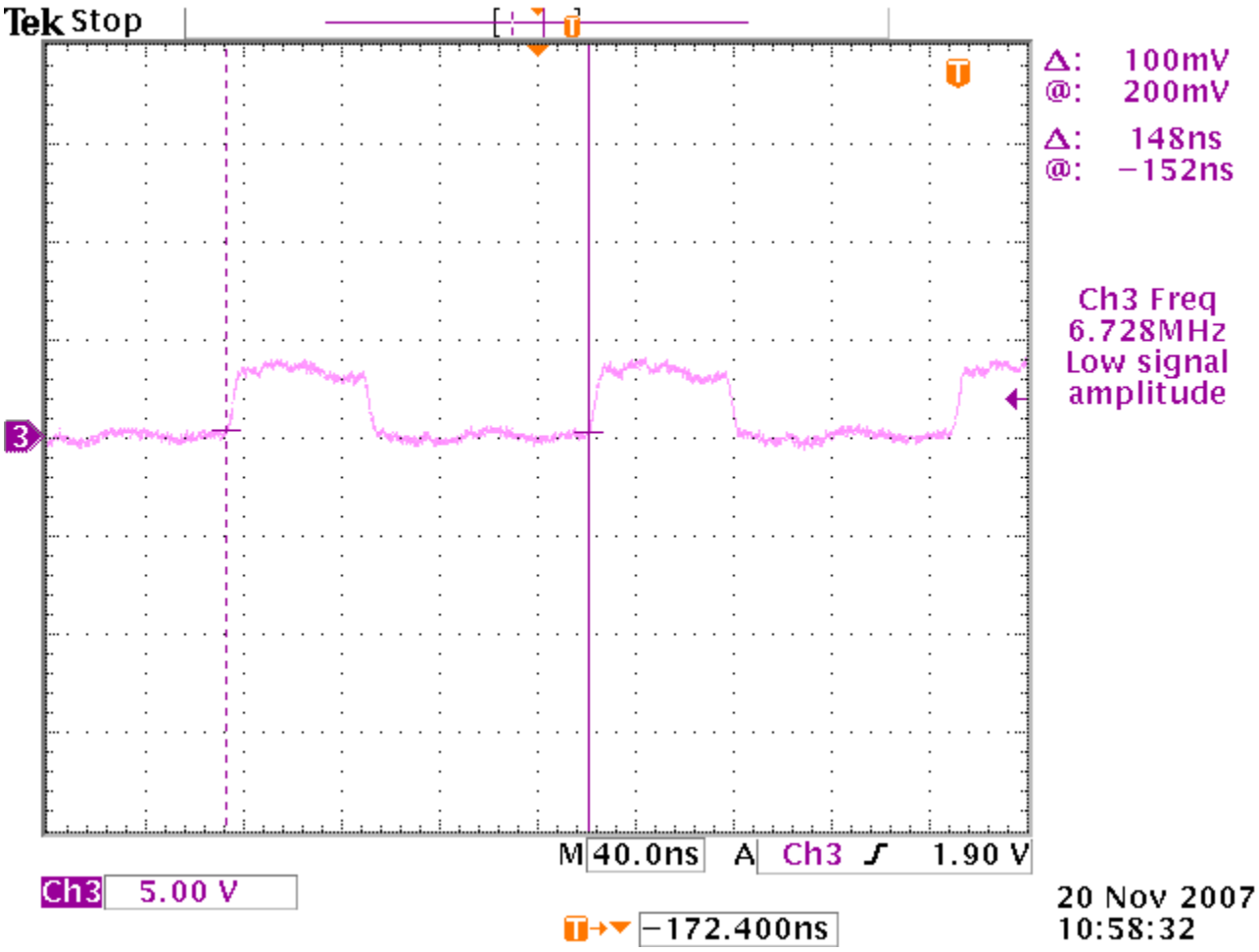


**Figure 1. RGPIO Set/Clear Square Wave**

In conclusion, the highest possible signal frequency on the Rapid GPIO output pin is 6.7 MHz (with the core operating at 50 MHz).

# 3 Timer

This section is focused on the output compare function of a timer module (TPM). With the output compare function, the TPM can generate timed pulses that have programmable position, polarity, duration, and frequency.

## 3.1 General Overview

This section contains selected passages from the MCF51QE128 reference manual that provide some details of the output compare function.

When the value in the timer counter register matches the channel value register, an interrupt flag bit is set and a selected output action is forced on the associated MCU pin. The output compare value may be selected to force the pin to zero, force the pin to one, toggle the pin, or ignore the pin (used for software timing functions).

The central component of the TPM is the 16-bit counter that can operate as a free running counter or a modulo up/down counter.

When a channel is configured for output compare (CPWMS=0, MSnB:MSnA=01 and ELSnB:ELSnA0≠00), the associated data direction control is overridden, the TPMxCHn pin is considered an output controlled by the TPM, and the ELSnB:ELSnA control bits determine how the pin is controlled.

When the output compare toggle mode is initially selected, the previous value on the pin is driven out until the next output compare event, then the pin is toggled.

TPMxCnSC[CHnF] — If the channel n is an output compare, CHnF is set when the value in the TPM counter registers matches the value in the TPM channel n value registers. Clear CHnF by reading TPMxCnSC while this bit is set and then writing a logic 0 to it.

In output compare mode, writing to either byte (TPMxCnVH or TPMxCnVL) latches the value into a buffer. After both bytes are written, they are transferred as a coherent 16-bit value into the timer-channel registers according to the value of CLKSB:CLKSA bits and the selected mode, so:

- If CLKSB and CLKSA are cleared, the registers are updated when the second byte is written.
- If CLKSB and CLKSA are not cleared and in output compare mode, the registers are updated after the second byte is written and on the next change of the TPM counter (end of the prescaler counting).

## 3.2 TPM Output Compare Application Example

This example shows output compare set/clear pin actions on a timer module channel.

```
TPM1MODH = 0x00;                    /* Period value setting */
TPM1MODL = 0x22;

(void)(TPM1C0SC == 0);/* Int. flag clearing (first part) */
TPM1C0SC = 0x18;      /* Int. flag clearing (2nd part) and configure pin action */
TPM1C0VH = 0x00; /* schedule pin action */
TPM1C0VL = 0x01;
TPM1SC = 0x08; /* enable input clock */

//main loop will repeat indefinitely
while (1)
{
      //wait until OC event occurs
      while (1) {
          if (TPM1C0SC & 0x80) {   //read OC event flag until it is set
              TPM1C0SC = 0x1C;     //clear OC flag, schedule next pin action
              TPM1C0VH = 0x00;
              TPM1C0VL = 0x11;
```

```
            break;      //exit while loop
        }
    }

    //wait until OC event occurs
    while (1) {
        if (TPM1C0SC & 0x80) {  //read OC event flag until it is set
            TPM1C0SC = 0x18;    //clear OC flag, schedule next pin action
            TPM1C0VH = 0x00;
            TPM1C0VL = 0x22;
            break;  //exit while loop
        }
    }
}
```

The same functionality can be achieved with toggle pin action with this code.

```
TPM1MODH = 0x00;                    /* Period value setting */
TPM1MODL = 0x11;

TPM1C0SC = 0x14; /* schedule next pin action - toggle */
TPM1C0VH = 0x00;
TPM1C0VL = 0x11;
TPM1SC = 0x08;  /* enable input clock */
```

**NOTE**

When output compare pin action is configured and scheduled, it does not need output compare flag clearing to happen again.

# 4 General-Purpose Bus Application Example

In this section we will describe the source code example that implements a 16-bit general purpose bus on the ColdFire V1 in < 0.6 kB of code size.

We assume a 50 MHz V1 core and a 25 MHz internal peripherals clock frequency. Suggested connections are found in Section 4.3, “Suggested Connections.”

**NOTE**

The functionality of the software example was validated by checking the generated signals with a scope. The functionality was not tested in connection with any kind of external device.

## 4.1 Output Compare Settings

- In this application example we use CLKSB:CLKSA = 01 (bus rate clock is the source of the TPM clock).
- Counter modulo register TPMxMODH:TPMxMODL = 0x0 (free running).
- The example code does not use TPMn interrupts (output compare events).

## 4.2 External Signals

This section describes the external signals involved in data-transfer operations.

**Table 1. External Signals**

| Signal Name | Direction | Description | Mapped for use as |
|---|---|---|---|
| RGPIO[15:0] | I/O | Multiplexed Address and Data Bus | AD[15:0] |
| TPM1CH0/PTA0 | O | Transfer Start | $\overline{TS}$ |
| TPM1CH1/PTB5 | O | Read Write | R/$\overline{W}$ |
| TPM2CH0/PTA1 | O | Chip Select | $\overline{CS}$ |
| TPM2CH1/PTB4 | O | Output Enable | $\overline{OE}$ |

## 4.3 Suggested Connections

This section shows possible connections for the general-purpose bus application example.

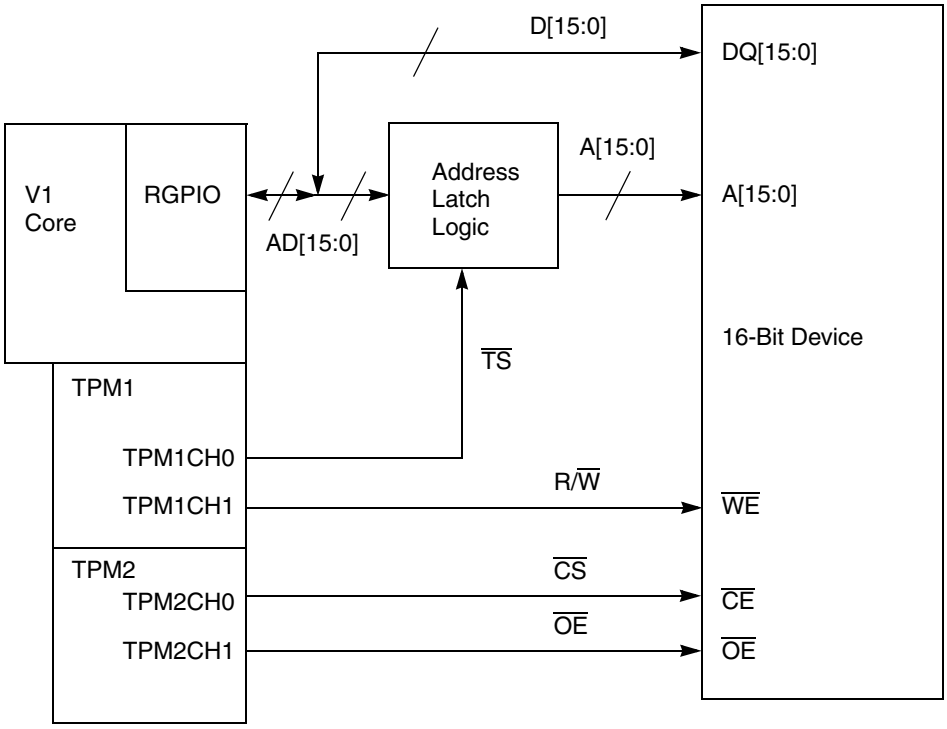


**Figure 2. Suggested General-Purpose Bus Connections**

The address latch logic block could in this case be a 16-bit negative-edge-triggered D-type flip-flop.

A 16-bit device could be an external memory, such as Freescale MRAM MR2A16A or Spansion flash S29AL016D.

## 4.4 Example Description

In this example, the functions below are implemented:

```
void mFxb_Init(void);
```

Initialization of TPMn modules, RGPIO module, and GPIO module.

```
void mFxb_QueuedRead(word* pAddr, word* pData, byte nTransfer);
```

Perform nTransfer read cycles with addresses from an address array — store read data in a data array.

```
void mFxb_QueuedWrite(word* pAddr, word* pData, byte nTransfer);
```

Perform nTransfer write cycles with address from an address array and with data from a data array.

### 4.4.1 mFxb_Init

This function will initialize the Rapid GPIO module, GPIO module, and timer/PWM module for the required functionality. The commented source code is provided below.

- Rapid GPIO module initialization:

```
RGPIO_ENB = 0xFFFF;//enable all RapidGPIO pins

RGPIO_DATA = 0x00;//drive all pins low
PTEPE = 0xFF;   //enable pull-ups
PTCPE = 0xFF;
RGPIO_DIR = 0xFFFF;//all RapidGPIO pins out direction
```

- GPIO module initialization
  TPM1 and TPM2 channel pins are multiplexed with GPIO module. When the TPMn clocks are stopped or no timer function is selected, the GPIO module will get control of the TPMn channel pins. Therefore we initialize GPIO to drive the output pins high.

```
PTBD |= (unsigned char)0x30;// PTB5 | PTB4
PTBDD |= (unsigned char)0x30; // data direction out
PTAD |= (unsigned char)0x43;// PTA6 | PTA1 | PTA0
PTADD |= (unsigned char)0x43;// data direction out
```

- TPMn module initializations for output compare on all TPMn channels, which are used to control the general-purpose bus — drive all channels high.

```
TPM1SC = 0x00;          /* Stop */
TPM1MODH = 0xFF;        /* Period value setting */
TPM1MODL = 0xFF;

TPM2SC = 0x00;          /* Stop */
```

```
TPM2MODH = 0xFF;       /* Period value setting */
TPM2MODL = 0xFF;

//INITIAL OUTPUT COMPARE EVENT FOR ALL CONTROL SIGNALS, WHICH ARE DRIVEN BY
TPM1 AND TPM2 MODULE

(void)(TPM1C0SC == 0);/* Channel 0 int. flag clearing (first part) */
TPM1C0SC = 0x1C;      /* Int. flag clearing (2nd part) and channel 0 contr.
register setting – OC event will set the channel pin */

/* First output compare value 0x0001 */
TPM1C0VH = 0x00;
TPM1C0VL = 0x01;

(void)(TPM1C1SC == 0);/* Channel 1 int. flag clearing (first part) */
TPM1C1SC = 0x1C;       /* Int. flag clearing (2nd part) and channel 1 contr.
register setting – OC event will set the channel pin */

/* First output compare value 0x0001 */
TPM1C1VH = 0x00;
TPM1C1VL = 0x01;

(void)(TPM1C2SC == 0);/* Channel 2 int. flag clearing (first part) */
TPM1C2SC = 0x1C;/* Int. flag clearing (2nd part) and channel 2 contr. register
setting – OC event will set the channel pin */

/* First output compare value 0x0001 */
TPM1C2VH = 0x00;
TPM1C2VL = 0x01;

(void)(TPM2C0SC == 0);// Channel 0 int. flag clearing (first part)
TPM2C0SC = 0x1C; /* – OC event will set the channel pin */

  /* First output compare value 0x0001 */
TPM2C0VH = 0x00;
TPM2C0VL = 0x01;



(void)(TPM1SC == 0);/* Overflow int. flag clearing (first part) */
```

```
(void)(TPM2SC == 0);/* Overflow int. flag clearing (first part) */

/* Enable clocks to TPM1 and TPM2 */
TPM1SC = 0x08;   /* Int. flag clearing (2nd part) and timer control register
setting */
TPM2SC = 0x08;/* Int. flag clearing (2nd part) and timer control register
setting */

//wait for 1st OC event on all channels
//by checking the CHnF flag
while (1)
{
    if(TPM1C0SC & 0x80)
    {
        TPM1SC = 0x00;//stop TPM1 clocks
        break;
    }
}
while (1)
{
    if(TPM2C0SC & 0x80)
    {
        TPM2SC = 0x00;//stop TPM2 clocks
        break;
    }
}

/* clear both counters */
TPM1CNT = 0x00;
TPM2CNT = 0x00;
```

### 4.4.2 mFxb_QueuedWrite

This function will control TPMnCHx pins, along with RGPIO pins, to perform a write cycle. The write cycle is repeated until the number of transfers is reached. Each write cycle has dedicated address and data parameters.

Figure 3 shows the function state flowchart.

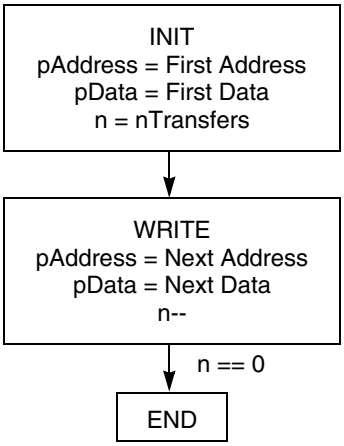


**Figure 3. QueuedWrite — State Flowchart**

A write cycle will control the Rapid GPIO module and TPM1 and TPM2 modules as seen in Figure 4:

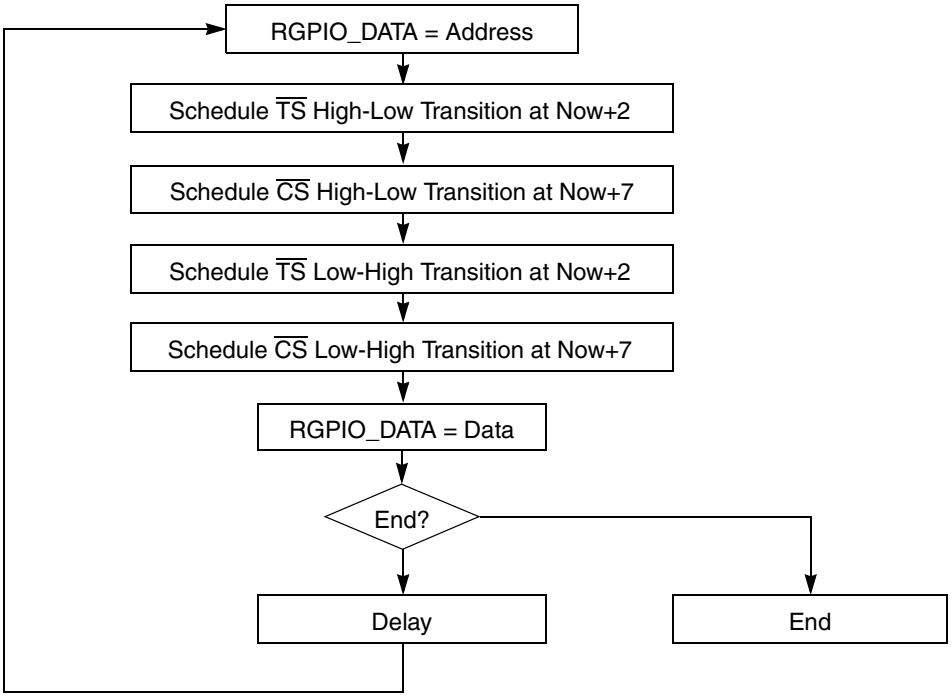


**Figure 4. QueuedWrite — Write Cycle Execution**

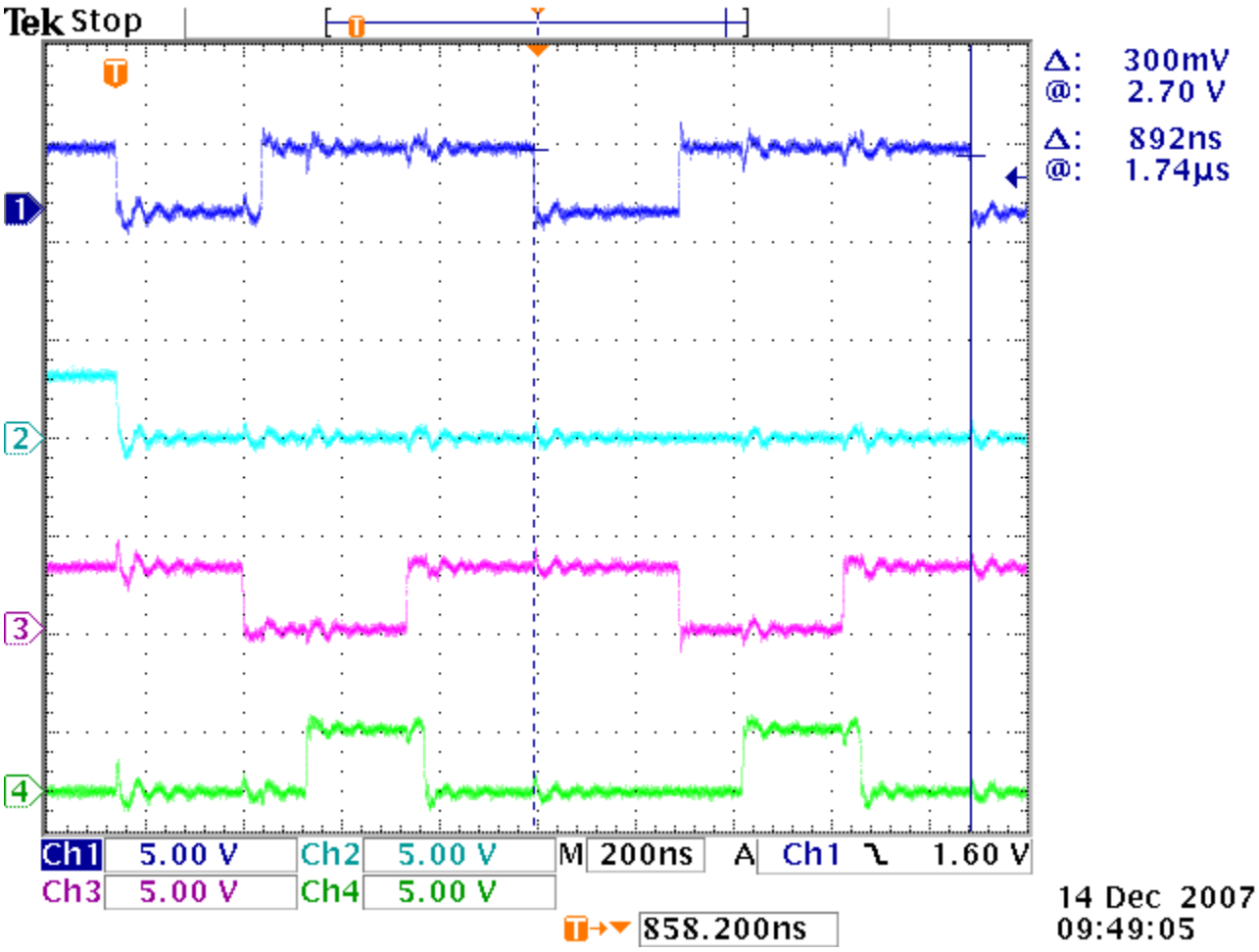


**Figure 5. Write Cycle — 1 $\overline{TS}$, 2 R/$\overline{W}$, 3 $\overline{CS}$, 4 RGPIO[0]**

### 4.4.3 mFxb_QueuedRead

This function will control TPMnCHx pins along with RGPIO pins to perform a read cycle. The read cycle is repeated until the number of transfers is reached. Each read cycle has dedicated address and data parameters.

Figure 6 shows the QueuedRead state flowchart:

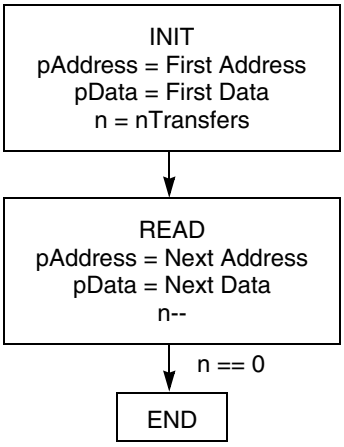


**Figure 6. QueuedRead — State Flowchart**

The read cycle will control the Rapid GPIO module and TPM1 and TPM2 module as seen in Figure 7:

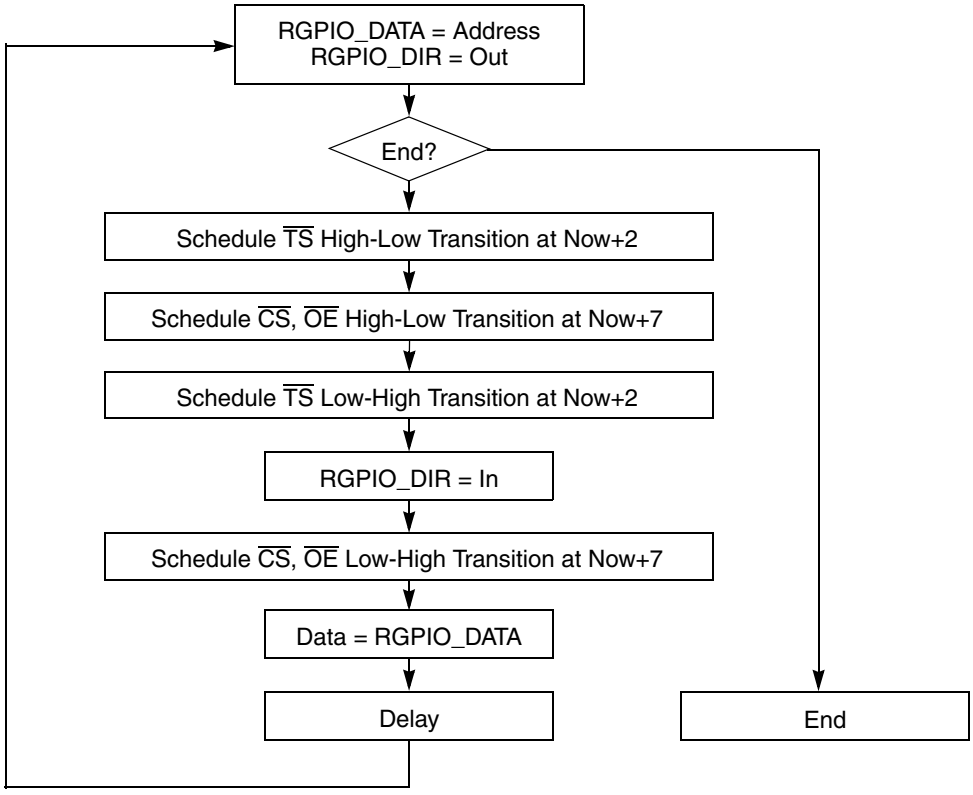


**Figure 7. QueuedRead — Read Cycle Execution**

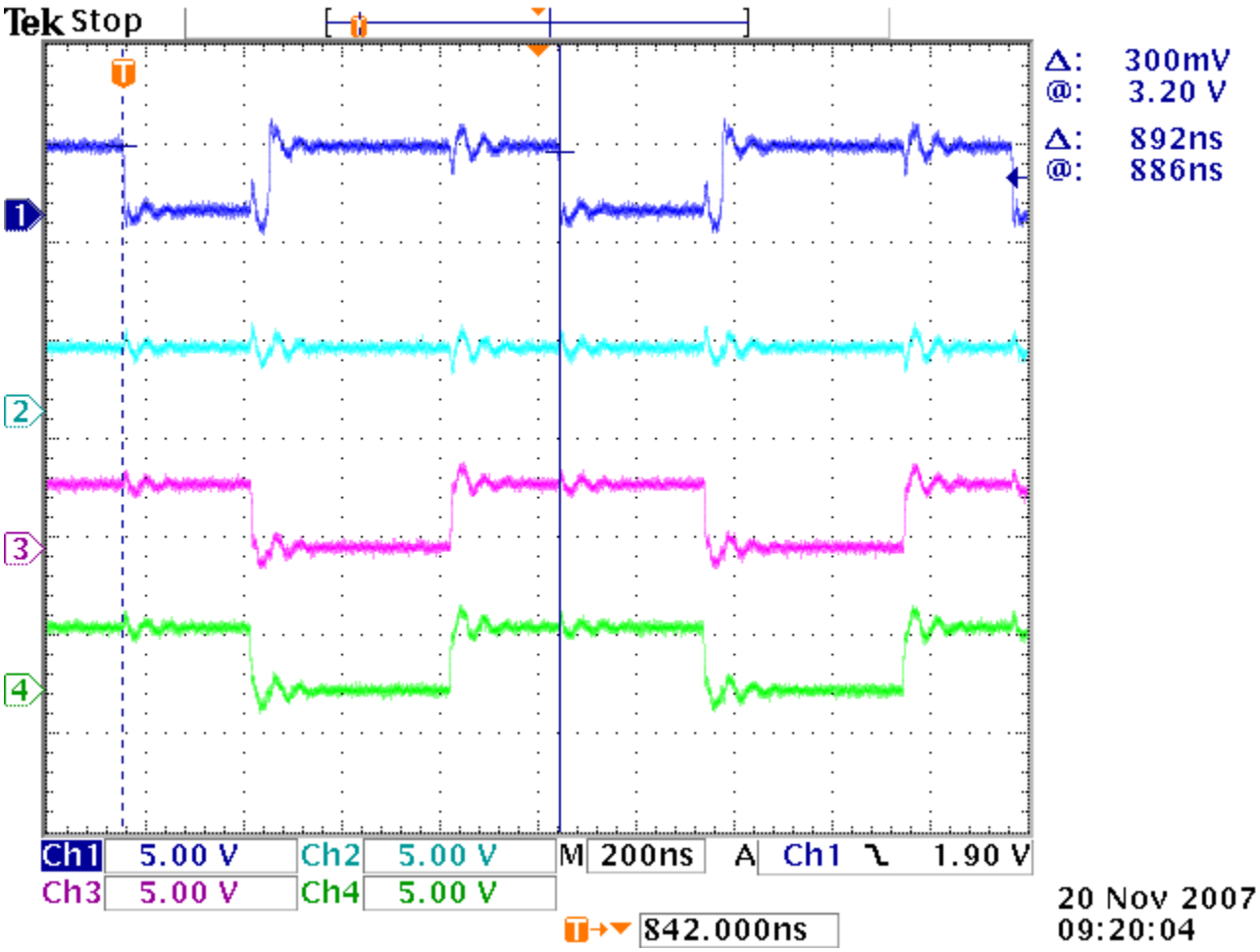


**Figure 8. Read Cycle — 1 #TS, 2 R/#W, 3 #CS, 4 #OE**

### 4.4.4 Use of Functions

This section describes how the functions may be used in a program.

```
volatile word addr[5];
volatile word data[5];

mFxb_Init();

addr[0] = 0x555;
data[0] = 0xAAA;

addr[1] = 0x2AA;
data[1] = 0x555;

addr[2] = 0x555;
```

```
data[2] = 0x0A0;

addr[3] = 0x0000;
addr[3] = 0x1234;

mFxb_QueuedWrite(&addr[0], &data[0], 4);

addr[0] = 0x0000;
mFxb_QueuedRead(&addr[0], &data[0], 1);
```

# 5 Conclusion

The Rapid GPIO module resides on the high-speed core bus, which allows it to generate digital signals with up to 6.7 MHz switching frequency.

The TPM output compare function may be used to generate timed pulses that have programmable position, polarity, duration, and frequency. The central component of a timer module is a 16-bit timer. Position of a pin action in time is configured by timer value register TPMxCnV (it compares with the actual count of the 16-bit timer). The pin action may be set, clear, toggle, or none.

Use of both modules is demonstrated by software implementation of a general-purpose multiplexed address/data bus.

Write and read cycle duration is less than 950 ns.

# 6 Testing and Validation

The software accompanying this application note was built around the DEMOQE128 board with MCF51QE128 running at 50 MHz frequency, using CodeWarrior for MCUs 6.1 IDE.

The software functionality was validated by measuring the relevant signals with an oscilloscope. Copies of the oscilloscope images are included in this document.

# 7 References

The MCF51QE128 reference manual provides detailed information about the MCF51QE128 family processors, and is available at www.freescale.com.

THIS PAGE IS INTENTIONALLY BLANK

**How to Reach Us:**

**Home Page:**
www.freescale.com

**Web Support:**
http://www.freescale.com/support

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3628
Rev. 0
04/2008



freescale™
semiconductor