Freescale Semiconductor
Application Note

Document Number: AN5240
Rev. 0, 01/2016

Reaction Module 2 for Peak&Hold Injection Control on the MPC5746R Using REACM2 Utility Functions

by: Marketa Venclikova

Contents

1 Introduction

This application note describes the example of Peak&Hold injection control utilizing a dedicated peripheral called the Reaction Module 2 (REACM2). The REACM2 features 10 output channels, each having three output pins that together with the Analogue to Digital convertor (ADC), Body Cross Triggering Unit (BCTU) and the enhanced Timer Processing Unit (eTPU) provide closed loop control functionality, REACM2 drives output signals based on ADC results delivered by BCTU and uses eTPU for timing. Sample code to drive the REACM2 on MPC5746R is associated with this application note. This sample code benefits from using general utility functions for REACM2 (AN5235) which ease the handling of the REACM2 module.

2 Functional overview

The REACM2 module comprises 10 Reaction Channels, each having three output pins. The REACM2 module is designed to allow closed feedback control by providing a modulation signal to control the solenoid injector or the valve current. The REACM2 has the following features:

- Independent output control architecture per channel
- Three outputs per channel to support different driver architectures

- Interface with the on-chip SAR ADC for fast response times
- Interface to eTPU and eMIOS
- Shared Modulation Control bank
- Flexible modulation settings based on timing and thresholding

The Reaction Channels are controlled by a Modulation Control Word provided by the Modulation Control Bank that is assigned to a respective channel. The Modulation Word provides information about the type of modulation to be executed as well as the addresses for the Threshold and Timer banks. The address of the first Modulation Control Word to be executed after the channel is enabled is stored in the channel.

REACM2 channels respond to timer signals from the timer inputs (eTPU, eMIOS) and on-chip ADC results coming to REACM2 through BCTU. Modulation on the channel is only executed if the channel is enabled, and time window on the respective timer input occurs. Each channel supports four modulation modes. Detail information can be found in the reference manual of the particular devices equipped with REACM2 module.

The Reaction Module 2 is composed of the following sub-modules:

- Reaction channel
- Modulation Control Word Bank
- Shared Timer Bank
- Hold Timer Bank
- Threshold Bank
- Period Pulse Generator

The figure below describes interconnection of the Reaction Module 2 sub-modules. Indexes on the picture indicate the consecutive steps in which each module executes its action during the modulation process:

1. A new ADC result from the BCTU module is received, triggering the respective Reaction Module Channel.
2. The triggered channel generates the address of the Modulation Word to be executed, based on the data programmed in the internal configuration register.
3. The respective Modulation Word is selected by the channel.
4. The selected Modulation Word generates the address for the Threshold Bank.
5. The threshold value selected by the Modulation Word is compared against the incoming ADC result.
6. The result of the comparison is sent back to the channel.
7. The channel executes the modulation action and updates the channel output.

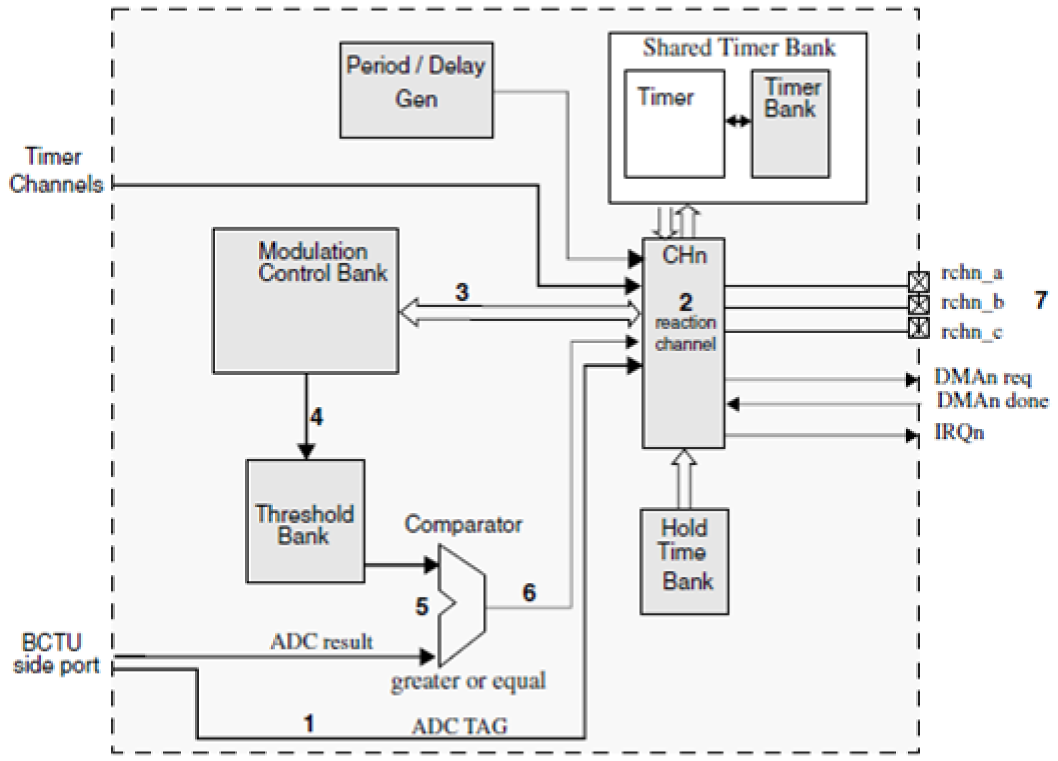

Figure 1. Interconnection of sub-modules within the Reaction Module 2

3 Application overview

This application is designed to control four fuel injectors configured as two separate banks, having two injectors per bank. Each bank is treated separately – independent of each other. The injection control application utilizes the following modules:

- Reaction Module 2 (REACM2) – two channels (one channel per bank)
- Enhanced Timer Processing unit eTPU – six channels (three channels per bank – two channel for injectors and one channel for the REACM2 channel - for time window generation)
- Body Cross-triggering Unit BCTU
- SAR Analog-to-Digital Converter – two channels (one channel per bank)

The figure below shows how each of the aforementioned modules interacts within this application. The figure presents connection for one bank of injectors (coils L1 and L2 represents solenoid injectors).

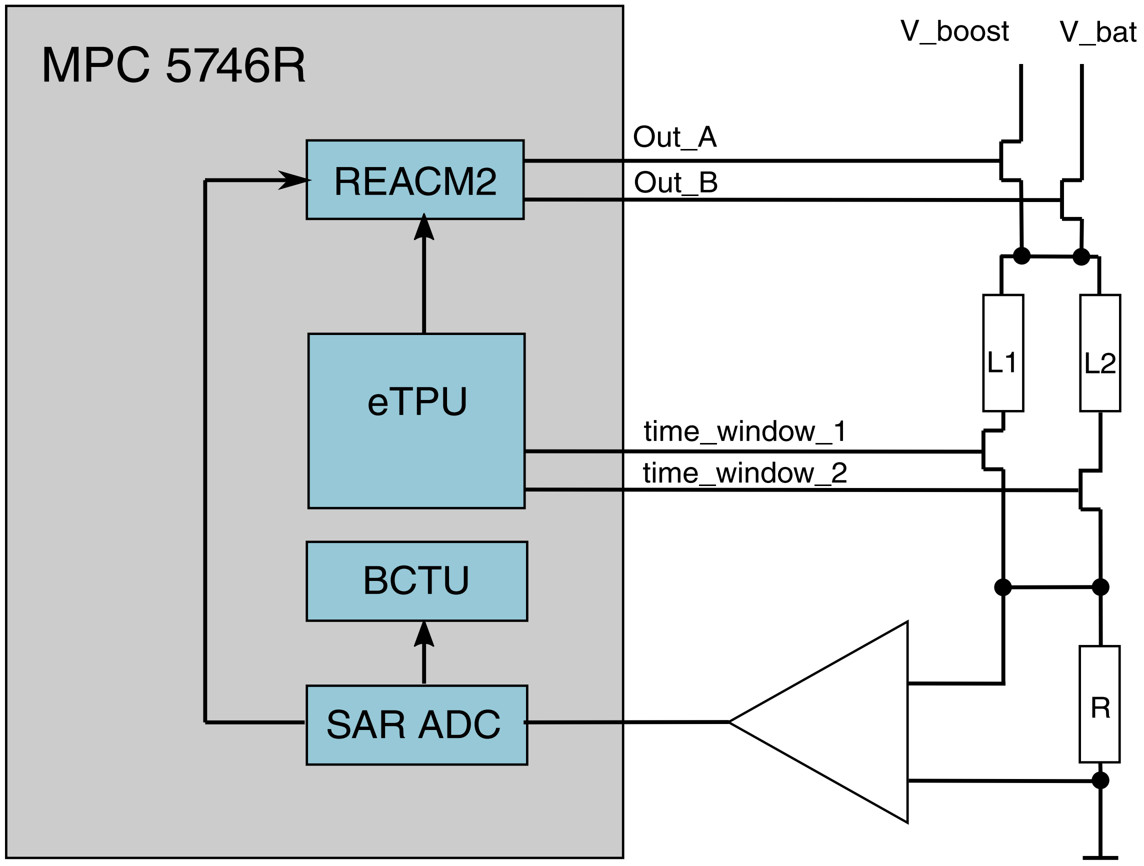

Figure 2. Connection of one injector bank

3.1 High-side voltage

There are two voltage sources: V_{bat} and V_{boost}. V_{boost} is a high voltage source of value 90 - 120 V. This voltage is switched into circuit to quickly open the injector and create Peak phase of the current waveform. V_{bat} is 12 V and is switched on after the Peak phase to keep injector open. This phase is called Hold phase.

3.2 High-side switches

One channel of the REACM2 drives two outputs, A and B (output C is unimplemented). The modulation on this channel causes FETs to switch controlling the high-side voltage, V_{bat} and V_{boost}. Switching of the high-side voltage controls the current flow into the injector's solenoid and thus its opening. Both high-side voltage sources are common for both injectors within a bank.

3.3 Current waveform

The current passing through an injector is captured using voltage measurements on the reference resistor R connected in series with the injectors. Voltage on this resistor is sensed with the on-chip ADC and the results passed to REACM2 through BCTU provide feedback to REACM2 channel.

In this application the aim is to control the Peak&Hold current characteristics passing through the solenoids, see the figure below.

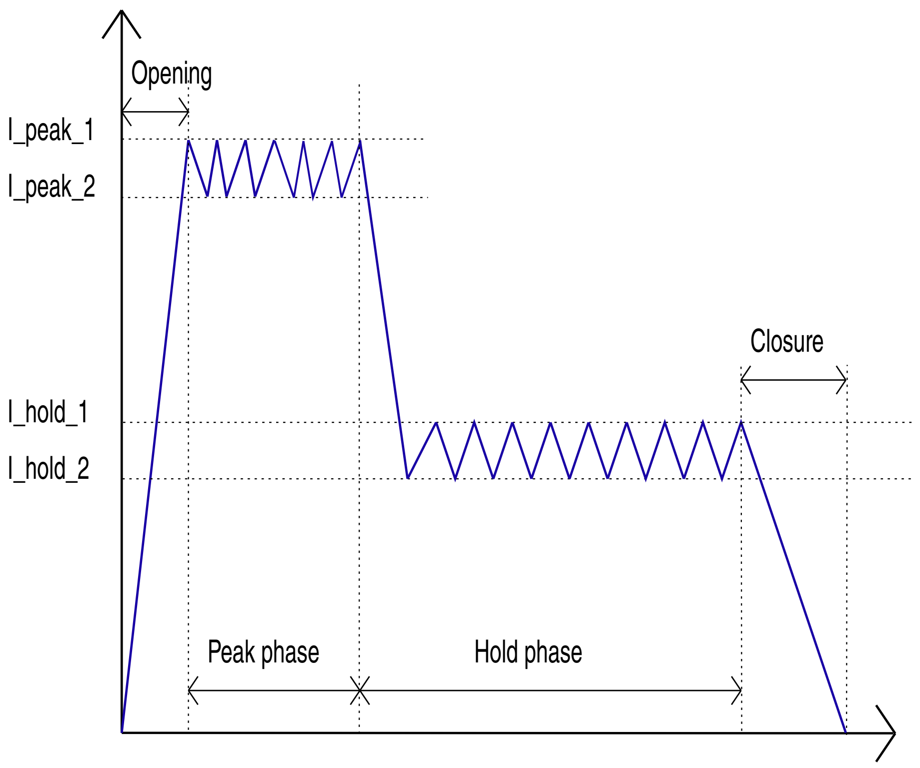

Figure 3. Peak&Hold current modulation

First phase of the waveform is the Opening of the injector. The current waveform in this phase should be steep enough to allow fast opening of the solenoid injector, thus the V_{boost} voltage source is switched. As soon as the current reaches I_peak_t1 threshold, it’s value is maintained between I_peak_t1 and I_peak_t2 thresholds for a specific time – this phase is called the Peak phase. This phase fully opens the injector. The Hold phase follows which keeps the injector open, requiring a lower current through the injector and thus switching lower voltage supply V_{bat}. Current values are maintained between I_hold_t1and I_hold_t2 thresholds here. Duration of the Hold phase is not fixed and is dependent on the respective time window signal that determines duration of the whole modulation cycle. The time window signal is generated by the eTPU using the Direct Injection function (see AN4907). The modulation is only executed during the time window signal is active on the respective timer input of the REACM2.One time window signal is needed per channel and only one modulation scheme can be executed at a time by the channel.

3.4 Low side switches

The eTPU also generates cylinder signals using the Knock function (see AN4907), which results in switching of the respective injector into the circuit. There are four cylinder signals in this application, one signal per injector. One bank utilizes one REACM2 channel, so the modulation performed within that one bank can be executed for only one injector at one particular time. But the modulation for injectors of different banks can overlap, see the figure below.

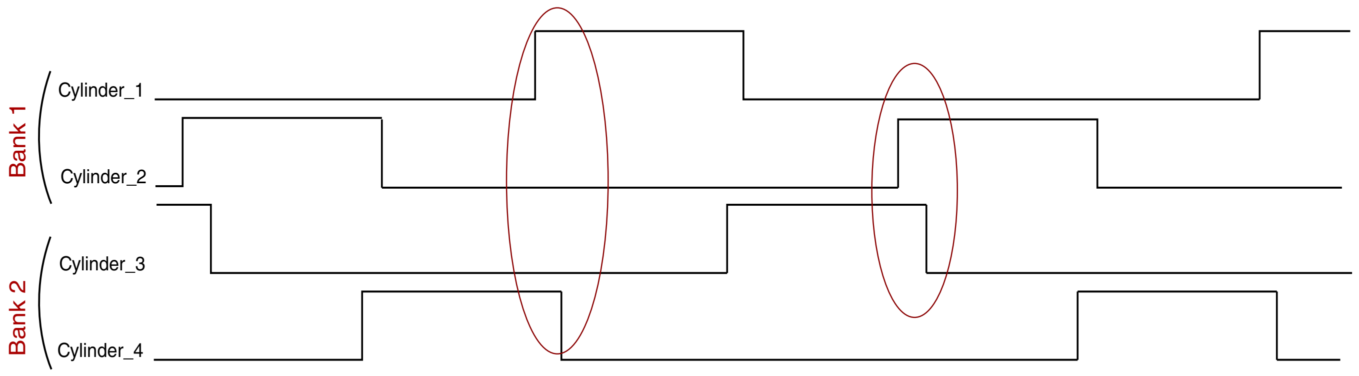

Figure 4. Cylinder signals for 2 banks, overlaps within banks are marked with red elipse

4 REACM2 general C utility functions

There are a set of general C functions for REACM2 reacm2_util.c and reacm2_util.h available to download together with example code reacm2_example.c showing how to use them (see AN5235). The REACM2 utility routines provide an easy way to set up REACM2 module and perform modulation on a channel, e.g. Peak&Hold current modulation. Utilization of these routines prevents the user need to access the REACM2 registers directly and eases REACM2 set up.The following routines are necessary to set up a modulation scheme running on a reaction channel:

```
fs_reacm2_module_init (REACM2_INIT_T * p_reacm2_init)

fs_reacm2_modword_set (MODULATION_WORD_T * phases,
                       uint8_t num_of_phases)

fs_reacm2_channel_init ( uint8_t ch_num,
                         REACM2_CHANNEL_INIT_T *p_channel_init,
                         uint8_t modulation_adress)

fs_reacm2_channel_enable (uint8_t ch_num,
                          REACM2_CHANNEL_INIT_T *p_channel_init )
```

4.1 Module initialization

The user is required to create a global initialization structure to define the general parameters of REACM2 module and its sub modules prior to the module initialization. The structure holds the configuration of the general module registers to be set:

```
typedef struct {
      uint32_t mcr;
      uint32_t tcr;
      uint32_t thrr;
      uint32_t pcr;
      uint32_t pscr;
      uint32_t adcmax;
      uint32_t range_pwd;
      uint32_t min_pwd;}
REACM2_INIT_T;
```

- mcr (uint32_t): configuration of the Module Configuration register. This register contains the control bits to set up general operation of the REACM2 module.
- tcr (uint32_t): configuration of the Timer Configuration Register. This register contains the prescaler configuration to set operation of the Hold Timer and Shared Timer.

- thrr (uint32_t): configuration of the Threshold Router Register. This register routes the ADC result to the Threshold bank. Setting of result routing is optional.
- pcr (uint32_t): configuration of the Period Generator Configuration Register. This register is used to setup the period pulse generator sub-block. This register must be configured when running Threshold-period modulation mode.
- pscr (uint32_t): configuration of the Period Shift Delay Configuration Register. This register is used to set up the delay between period pulses of two consecutive channels.
- adcmax (uint32_t): ADC Result Maximum Limit Check register. This register holds the maximum expected value of the ADC result.
- range_pwd (uint32_t): Modulation Range Pulse Width register . This register provides the value used for the PWM pulse width check during the modulation process.
- min_pwd (uint32_t): Modulation Minimum Pulse Width register. This register provides the value used to check whether the pulse width generated during the PWM modulation process on channel output is shorter than a minimum pulse width.

A more detailed description of the general module registers can be found in the reference manual of the respective device equipped with a REACM2.

The routine fs_reacm2_module_init()performs module initialization, having a pointer to the user pre-initialized structure as an input parameter.

4.2 Modulation Control Word

Before the channel starts to execute modulation on its output, the Modulation Control Word which defines the parameters of the modulation must be stored into the Modulation Control Word Bank. The Modulation Control Word Bank is a set of registers that control the Reaction Channel Modulation scheme. The registers are initialized by software and read by the Reaction channels. The Modulation Words comprise information required by the Reaction Channels to execute modulation. All the Reaction Channels have access to the same words, the Modulation Control Word Bank is shared within Reaction Module 2. It means that multiple Reaction Channels can perform the same modulation at the same time. Threshold and timer values are also stored in their respective banks of the REACM2 module, namely Threshold Bank, Shared Timer Bank, and Hold Timer Bank. These banks are shared within the REACM2 too, so that multiple Modulation Control Words can store the index of one threshold or timer value, respectively.

Prior to performing storage of the Modulation Control word with REACM2 utility routines, the user is required to define several structures containing parameters of the modulation:

- All the thresholds and timer values have to be defined in form of their respective structures (THRESHOLD_T, SH_TIME_T,HOLD_TIME_T) as thrs, sh_time or hold_time values respectively. The user is strongly recommended to fill idx with NAN value and let it be calculated automatically by software.
- Definition of the MODULATION_WORD_T structure comprising modulation specification and pointers to the aforementioned time and threshold structures. If there are any time or threshold pointers unused in the modulation word structure, the user is required to fill them with NULL pointer.
- If multiple Modulation Control Word storage is required, user is advised to create an array of MODULATION_WORD_T structures and pass a pointer to this array into the fs_reacm2_modword_set() routine.

The figure below shows the user defined structures of the Modulation Control Word and their utilization while setting the modulation word. The routine fs_reacm2_modword_set() stores threshold and timer values into threshold and timer banks, it then assigns indexes corresponding to their location within the Modulation Control Word. It means that the Modulation Control Word holds indexes corresponding to the location of the threshold and time values in the banks.

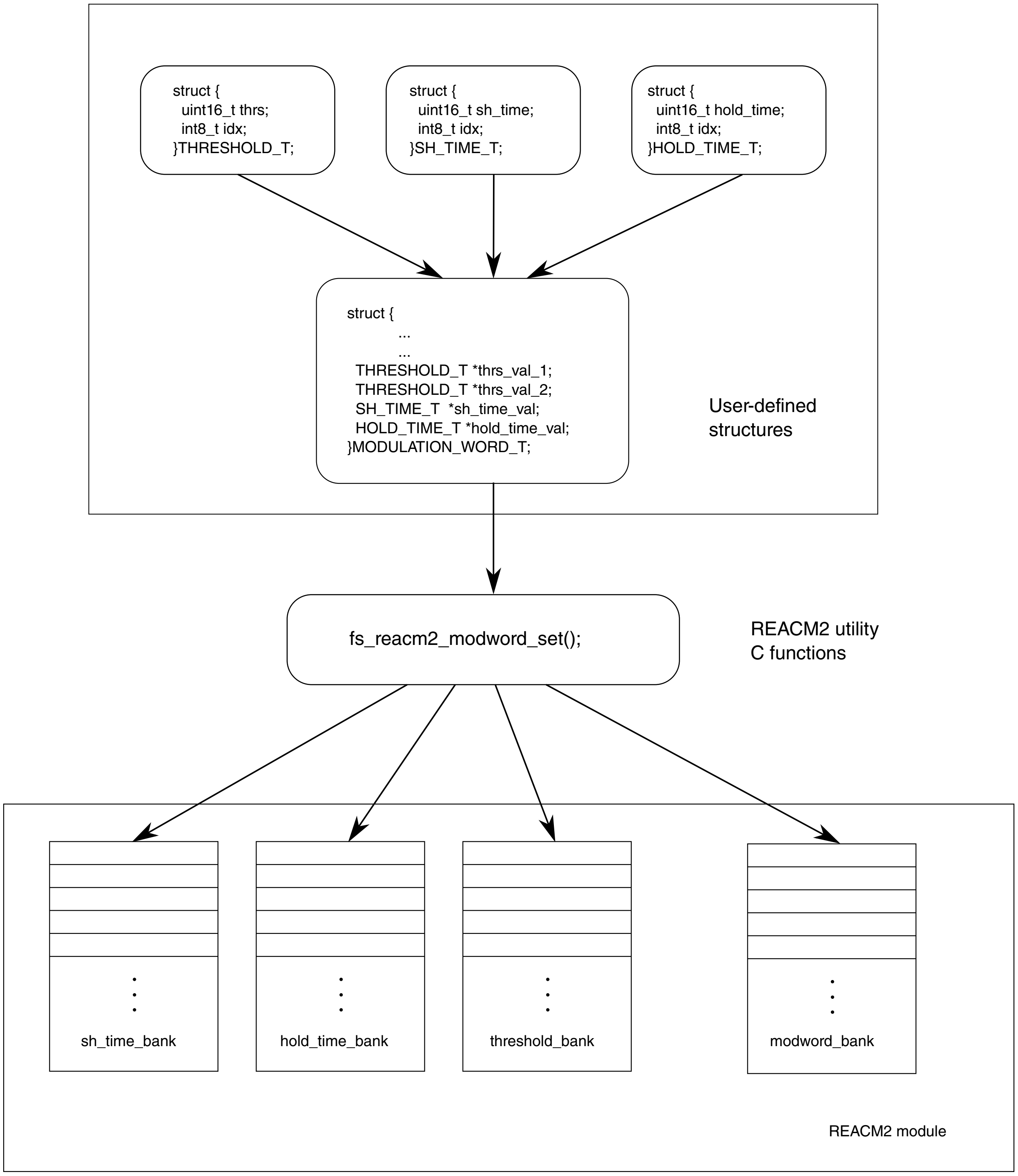

Figure 5. Modulation Control Word settings and structures to define modulation parameters

4.2.1 Modulation Control Word structure

Definition of the Modulation Control Words must be done as an array of structures type MODULATION_WORD_T prior to the modulation word being stored in the Modulation Control Word Bank. The structure is defined as follows:

```
struct {
      uint32_t loopback;
      uint32_t init_output_val;
      uint32_t mod_mode;
      uint32_t seq_mode;
      uint32_t hod_c;
      uint32_t hod_b;
      uint32_t hod_a;
      uint32_t lod_c;
      uint32_t lod_b;
      uint32_t lod_a;
      THRESHOLD_T *thrs_val_1;
      THRESHOLD_T *thrs_val_2;
        SH_TIME_T   *sh_time_val;
      HOLD_TIME_T *hold_time_val;}
MODULATION_WORD_T;
```

- loopback (uint32_t): LOOP control bit settings. This field indicates whether the next Modulation Control Word accessed by the Reaction Channel after a Modulation Word address increment event occurs will be the initial Modulation Control Word for that particular channel. The following Modulation Control Word definition can be found in *reacm2_util.h:*
  - FS_REACM2_LOOP_ON
  - FS_REACM2_LOOP_OFF
- init_output_val (uint32_t): IOSS Initial Output State selection. This field defines the state of the Reaction Channel output pins after modulation has started on the channel. The following Modulation Control Word definition can be found in *reacm2_util.h*:
  - FS_REACM2_IOSS_LOD
  - FS_REACM2_IOSS_HOD
- mod_mode (uint32_t): MM modulation mode settings. This field indicates the modulation that is executed by the channel, while the IOSS determines the modulation mode initial state. The following Modulation Control Word definition can be found in *reacm2_util.h*:
  - FS_REACM2_MM_TT
  - FS_REACM2_MM_THOFF
  - FS_REACM2_MM_THON
  - FS_REACM2_MM_TLOWP
- Seq_mode (uint32_t): SM Sequencer mode settings. This field defines the event that makes the channel Modulation Word address increment to the next Modulation Word. The following Modulation Control Word definition can be found in *reacm2_util.h*:
  - FS_REACM2_SM_NO_TRANSITION
  - FS_REACM2_SM_TIMER_EVENT
  - FS_REACM2_SM_HOLD_TIMER_EVENT
  - FS_REACM2_SM_THRESHOLD_EVENT
- hod_a, hod_b, hod_c (uint32_t): High Output Drives. The HOD field defines the values driven on the output pins pin_a, pin_b and pin_c when the channel is in the ON state. The following Modulation Control Word definition can be found in *reacm2_util.h*:
  - FS_REACM2_HOD_A_LOW
  - FS_REACM2_HOD_A_HIGH
  - FS_REACM2_HOD_B_LOW
  - FS_REACM2_HOD_B_HIGH
  - FS_REACM2_HOD_C_LOW
  - FS_REACM2_HOD_C_HIGH
- lod_a, lod_b, lod_c (uint32_t): Low Output Drives. The LOD field defines the values driven on to the output pins pin_a, pin_b and pin_c when the channel is in the OFF state. The following Modulation Control Word definition can be found in *reacm2_util.h*:
  - FS_REACM2_LOD_A_LOW
  - FS_REACM2_LOD_A_HIGH
  - FS_REACM2_LOD_B_LOW
  - FS_REACM2_LOD_B_HIGH

    - FS_REACM2_LOD_C_LOW
    - FS_REACM2_LOD_C_HIGH
- *thrs_val_1, *thrs_val_2 (THRESHOLD_T): Pointers to a structure of type THRESHOLD_T comprising the value of the threshold to be stored into the Threshold Bank and index *idx* of the Threshold Bank, where the threshold value is stored. THRESHOLD_T structures have to be predefined by the user.

```
struct{
        uint16_t thrs;
        int8_t idx;
}THRESHOLD_T;
```

- *sh_time_val (SH_TIME_T): A pointer to the structure of type SH_TIME_T comprising the value of the Shared Timer to be stored into Shared Timer Bank and index *idx* of the Shared Timer Bank, where the *sh_time* value is stored. SH_TIME_Tstructures have to be predefined by the user.

```
struct{
       uint16_t sh_time;
       int8_t idx;
}SH_TIME_T;
```

- *hold_time_val (HOLD_TIME_T): A pointer to the structure of type HOLD_TIME_T comprising the value of the Hold Timer to be stored into the Hold Timer Bank and index *idx* of the Hold Timer Bank, where the *hold_time* value is stored. HOLD_TIME_Tstructures have to be predefined the by user.

```
struct{
       uint16_t hold_time;
       int8_t idx;
}HOLD_TIME_T;
```

4.2.2 Modulation Control Word storage

Modulation word storage is performed by calling the fs_reacm2_modword_set() routine. This routine performs the following steps:

- Performs a check of the availability on the Modulation Control Word Bank, Threshold Bank, Shared Timer Bank and Hold Timer Bank to store new defined thresholds and time values there
- Performs storage of the threshold and time values into their respective banks
- Performs settings of modulation parameters
- Automatically calculates indexes of the thresholds and time values for the modulation word
- Stores Modulation Control Words into the Modulation Control Word Bank
- Is able to store multiple consecutive Modulation Control Words

NOTE

If storage was successful the routine returns the index of the first Modulation Control Word stored within this routine, otherwise it returns an ERROR code.

4.3 Channel initialization

To initialize the Reaction Channel, the user is required to define REACM2_CHANNEL_INIT_T structure. Channel initialization is performed by calling the following routine:

```
fs_reacm2_channel_init    ( uint8_t ch_num,
                            REACM2_CHANNEL_INIT_T *p_channel_init,
                            uint8_t modulation_adress)
```

- ch_num (uint8_t): index of the channel that is to be initialized
- *p_channel_init (REACM2_CHANNEL_INIT_T): pointer to a user-defined channel initialization structure:

```
struct {
        uint32_t chcr;
        uint32_t chrr;
}REACM2_CHANNEL_INIT_T;
```

- chcr (uint32_t): Channel Configuration Register settings. This register controls channel behavior, including interrupt settings and the reaction channel output disable state.
- chrr (uint32_t): Channel Router Register settings. This register controls the channel connection to external timers (eTPU, eMIOS) and ADC result data.

- Modulation_address (uint8_t): index of the first Modulation Control Word in the Modulation Control Word Bank to be executed by the channel. This index is returned by the *fs_reacm2_modword_set()* routine. User is advised to call this routine prior calling the *fs_reacm2_channel_init()* routine to ensure that the particular Modulation Control Word is stored in Modulation Control Word Bank.

4.4 Channel enable – start performing modulation

After the REACM2 module and the particular reaction channel initialization is complete, modulation on the channel can be initiated by the fs_reacm2_channel_enable() routine. The channel is enabled either in timer or software control mode, depending on the Channel Control Register settings:

- SW control mode – channel starts to perform the modulation as soon as channel is enabled. The modulation is terminated when the channel is disabled.
- Timer control mode – when enabled in this mode, the channel performs the modulation only when the timer window signal is active on the respective timer REACM2 input.

5 Example code to set Peak&Hold modulation utilizing REACM2 utility functions

The following example code shows the utilization of *reacm2_util.c* and *reacm2_util.h* files to set up the REACM2 module and the channel to perform Peak&Hold current modulation for solenoid fuel injector control. In this application solenoid injectors are replaced with RC serial circuitry and there is only one voltage source V_bat utilized, V_boost is not implemented. V_bat is sourced from the MCU pin where the eTPU channel generates the *time window* signal. The following figure shows the schematics of the circuit.

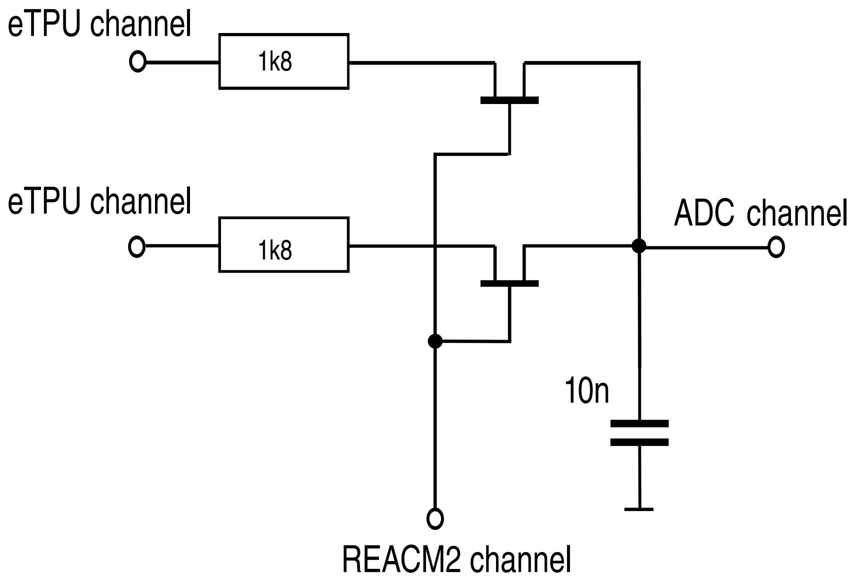

Figure 6. Schematic of the circuitry replacing one bank of solenoids

To create a Peak&Hold current modulation like shape four modulation words are defined. First, the module and channel initialization structures are defined, followed by the threshold and timer structure definitions. Then an array of four members type MODULATION_WORD_T to define four modulation words is created. After that the routines to initialize the Reaction Module and the Reaction Channel are called. As soon as at least one Modulation Control Word is stored in the Modulation Control Word Bank and the address of the first Modulation Control Word is assigned to the channel, the channel is enabled and starts to output the modulated signal.

5.1 Example code

```
/*module initialization structure */
REACM2_INIT_T my_module_setup =
{
  /* MCR init */
  FS_REACM2_MDIS_ENABLE | FS_REACM2_TPREN_ENABLE | FS_REACM2_HPREN_ENABLE,
  /* TCR init */
  FS_REACM2_HPRE(2) | FS_REACM2_TPRE(2),
  /* THRR init */
  FS_REACM2_WREN1_DISABLE | FS_REACM2_WREN0_DISABLE,
  /* PCR init */
  FS_REACM2_PERPRESC(1) | FS_REACM2_PER(0), // period generator stopped
  /* PSCR init */
  FS_REACM2_DLY(0), // no delay between the period pulses of the consecutive channel
  /* ADCMAX init */
  FS_REACM2_ADCMAX(0), //when set to 0 the adc max limit checking disabled
  /* RANGE_PWD init */
  FS_REACM2_RANGE_PWD(0), // no pwd range checking is performed
  /* MIN_PWD init */
  FS_REACM2_MIN_PWD(0) // no minimum pulse width checking is performed
};
/* channel initialization structure */
REACM2_CHANNEL_INIT_T ch_bank1=
{  /* CHCR init */
   FS_REACM2_CHEN_ENABLE | FS_REACM2_DOFF_C_LOW | FS_REACM2_DOFF_B_LOW |
FS_REACM2_DOFF_A_LOW,
   /* CHRR init */
   FS_REACM2_ADCR(FS_REACM2_ADC_TAG_4)| FS_REACM2_CHIR(0)
};
REACM2_CHANNEL_INIT_T ch_bank2 =
{
   /* CHCR init */
   FS_REACM2_CHEN_ENABLE | FS_REACM2_DOFF_C_LOW | FS_REACM2_DOFF_B_LOW |
FS_REACM2_DOFF_A_LOW,
   /* CHRR init */
   FS_REACM2_ADCR(FS_REACM2_ADC_TAG_5)| FS_REACM2_CHIR(1)
};
/* threshold and time definitions */
   THRESHOLD_T thsB00 ={ 0xb00, NAN};
   THRESHOLD_T thsA00 ={ 0xa00, NAN};
   THRESHOLD_T ths800 ={ 0x800, NAN};
   THRESHOLD_T ths600 ={ 0x600, NAN}
   SH_TIME_T stm0E00 ={0x0e00, NAN};
   SH_TIME_T stm3000 ={0x3000, NAN};
   SH_TIME_T stm0250 ={0x0250, NAN};
/* modulation word parameters definition */
MODULATION_WORD_T my_phases [4]={
/* loop, ioss, mod, sequencer mode, HOD_C out, HOD_B out, HOD_A out, LOD_C out, LOD_B out,
LOD_A out, threshold 1, threshold 2, sh_time, hold_time */
       {FS_REACM2_LOOP_OFF, FS_REACM2_IOSS_LOD, FS_REACM2_MM_TT,
FS_REACM2_SM_THRESHOLD_EVENT, FS_REACM2_HOD_C_HIGH, FS_REACM2_HOD_B_HIGH,
FS_REACM2_HOD_A_HIGH, FS_REACM2_LOD_C_LOW, FS_REACM2_LOD_B_LOW, FS_REACM2_LOD_A_LOW,
&thsB00,&thsA00,&stm0E00,NULL},
        {FS_REACM2_LOOP_OFF, FS_REACM2_IOSS_LOD, FS_REACM2_MM_TT,
FS_REACM2_SM_TIMER_EVENT, FS_REACM2_HOD_C_HIGH, FS_REACM2_HOD_B_HIGH,
FS_REACM2_HOD_A_HIGH, FS_REACM2_LOD_C_LOW, FS_REACM2_LOD_B_LOW, FS_REACM2_LOD_A_LOW,
&thsB00,&thsA00,&stm3000,NULL},
        {FS_REACM2_LOOP_OFF, FS_REACM2_IOSS_LOD, FS_REACM2_MM_TT,
FS_REACM2_SM_TIMER_EVENT, FS_REACM2_HOD_C_HIGH, FS_REACM2_HOD_B_HIGH,
FS_REACM2_HOD_A_HIGH, FS_REACM2_LOD_C_LOW, FS_REACM2_LOD_B_LOW, FS_REACM2_LOD_A_LOW,
NULL,NULL,&stm0250,NULL},
        {FS_REACM2_LOOP_ON, FS_REACM2_IOSS_LOD, FS_REACM2_MM_TT,
FS_REACM2_SM_NO_TRANSITION, FS_REACM2_HOD_C_HIGH, FS_REACM2_HOD_B_HIGH,
FS_REACM2_HOD_A_HIGH, FS_REACM2_LOD_C_LOW, FS_REACM2_LOD_B_LOW, FS_REACM2_LOD_A_LOW,
&ths800,&ths600,NULL,NULL}};
void main()
{
   uint8_t address = 0;
   /* call functions that initialize the REACM2 module and channel */
   fs_reacm2_module_init(&my_module_setup);
   address = fs_reacm2_modword_set(  &my_phases,
                                     (sizeof(my_phases)/sizeof(MODULATION_WORD_T)));
   fs_reacm2_channel_init(FS_REACM2_CH_4,&ch_bank1, address);
   fs_reacm2_channel_init(FS_REACM2_CH_5,&ch_bank2, address);
   fs_reacm2_channel_enable(FS_REACM2_CH_4,&ch_bank1);
   fs_reacm2_channel_enable(FS_REACM2_CH_5,&ch_bank2);
   while(1)
   {
   }
}
```

5.2 Program output

The figure below shows the output from the Reaction Channel performing modulation (REACM2 ch2) and the corresponding current modulation (green line).

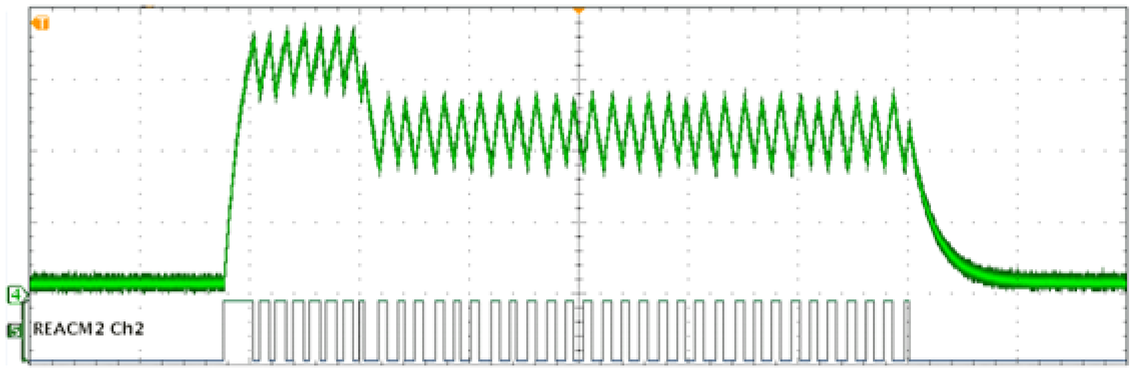

Figure 7. REACM2 output and current modulation in detail

In the figure below, all the outputs from Reaction Channels are shown together with their respective knock signals and injector current timing characteristics. It is noticeable that knock signals between two banks can overlap, but that overlap never occurs within one bank.

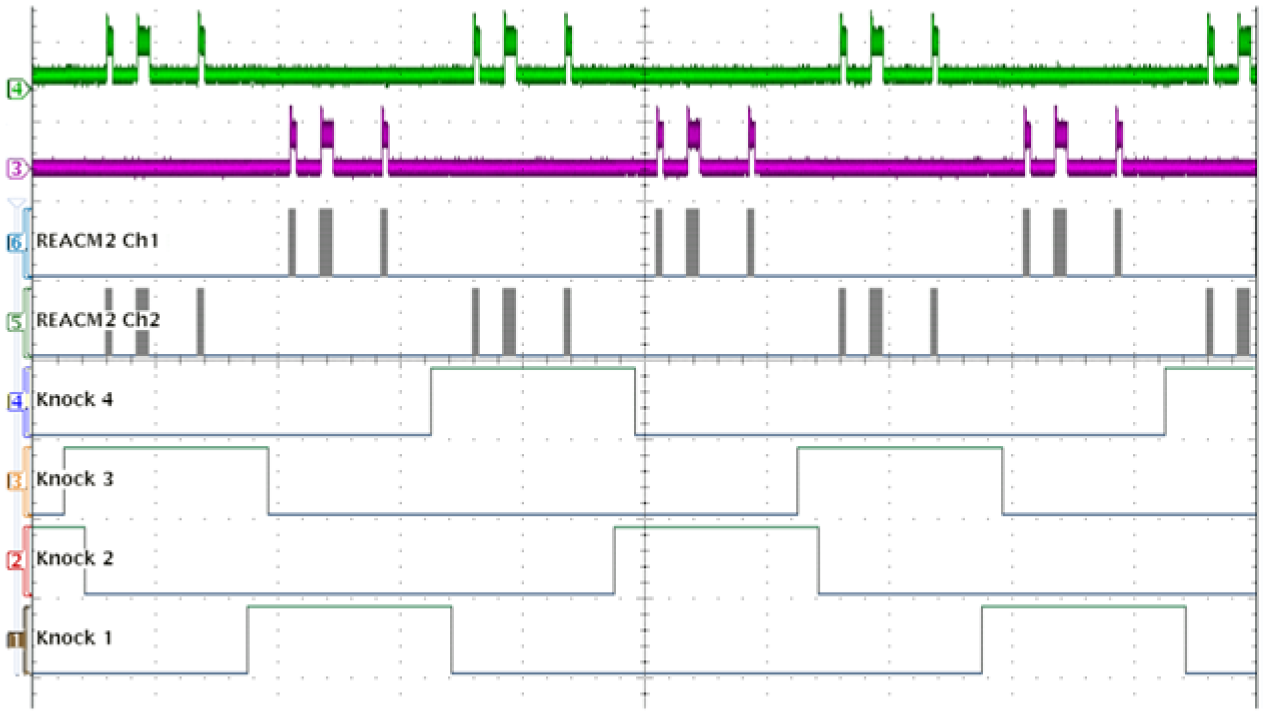

Figure 8. All the application outputs driving four injectors in two banks

6 Summary

This application note provides a description of the dedicated peripheral REACM2 and an example of its utilization to control solenoid fuel injectors. The general C REACM2 utility routines are used to set up the REACM2 module and perform Peak&Hold current modulation. This example is suitable for MPC5746R and all REACM2 equipped devices.

Document Number AN5240
Revision 0, 01/2016