Regtify Application Note 001 Dezember 13, 2004

Generating a Field Access Layer for Firmware Programming

Abstract: This document describes how Regtify can be used to generate field access functions for bitfields in assembler files. A field in this context is a group of consecutive bits within a register which controls a distinct function of a device. Registers are means to access these bitfields. From a programmer's view, fields are the objects of interests. Operating with masks and hardcoded bit positions is an inconvenient and not very readable way of dealing with bitfields. If registers and bitfields are defined within Regtify, it is easily possible to generate arbitrary access functions for different target languages. As a result, readability and transparency of the assembler code is significantly improved without sacrificing code efficiency. The implementation for use within the Altium Tasking 166/167 environment is provided as example.

Audience: This document is dedicated to developers and programmers using assembler languages, especially the ASM166.

Concept: Within the Tasking ASM166 Macro Assembler Language, it is possible to compare strings, use compile time variables and to write macro functions. Setfield and getfield functions will be constructed which perform the following tasks as checks for register field existence, masking and shifting the field values into CPU registers and so on.

Implementation Details:

The following functions may be used within the Assembler code:

Macroname	@SETFIELDV
Parameter	REGNAME,FIELDNAME,FIELDVALUE
Description	Sets the fieldbits of FIELDNAME of the internal variable @REGVAL to FIELDVALUE. @REGVAL can later be written to the actual register destination.

Macroname	@SETFIELDSV
Parameter	REGNAME,{FIELDNAME,FIELDVALUE} +
Description	Sets multiple fields by using only one macro call to internal variable @REGVAL to be later written to the actual register destination.

Macroname	@SETFIELD
Parameter	REGNAME,FIELDNAME,FIELDVALUE,TEMP
Description	Reads the value from the register and stores it in TEMP, set the fieldbits in TEMP and writes TEMP to the register.

Macroname	@GETFIELD
Parameter	TARGET,REGNAME,FIELDNAME
Description	Reads the value from the register into the target, masks out undesired fields and move the field value to the first bit position.

Macroname	@GETREG
Parameter	REGNAME,SOURCE
Description	Sets the register to the source

Macroname	@SETREG
Parameter	TARGET,REGNAME
Description	Reads from the register REGNAME and writes it to TARGET

Of course, also more functions can be composed. Depending on the address range, different assembler commands can be used, if certain operations are not feasible in the complete address range. E.g., special function registers may have bitaccess functionality whereas normal registers within the ordinary system address space can only be accessed completely. Furthermore, you can locate more easily the positions within your source code where fields are accessed which helps when changes are needed or functional errors have to be found.

How the functions are generated:

The **Regtify**-Driver Mechanism goes multiple times - named “passes” - through the complete database and executes the TCL-Code for each register defined in the database. Within a loop, the field attributes can be accessed. By checking for isFirstRegister and isLastRegister header and footer part of the functions can be written.

In the first pass, mask and positions are defined for each field in the design. This allows also the use of mask and positions attributes in the source code.
In the second pass, SETFIELDV is defined, which relies on the masks previously defined.

In pass three, SETFIELD is written out

In pass four, GETFIELD is generated and finally all functions which do not need to use register or field-specifics themselves are written out, like GETREG, SETREG and SETFIELDSV, which calls for each FIELDNAME-VALUE pair the SETFIELDV macro.

An example of macro definition output:

```
; ================================================================================
; Mask and Position Definitions
@SET(ADC0_CONTROL_CHANNEL000_MSK,0000000000000001b)
@SET(ADC0_CONTROL_CHANNEL000_POS,0)
@SET(ADC0_CONTROL_CHANNEL001_MSK,0000000000000010b)
@SET(ADC0_CONTROL_CHANNEL001_POS,1)
@SET(ADC0_CONTROL_CHANNEL002_MSK,0000000000000100b)
@SET(ADC0_CONTROL_CHANNEL002_POS,2)
; ================================================================================
@*DEFINE SETFIELDV(REGNAME,FIELDNAME,VALUE)

@SET(FOUND,0)
@IF ( @EQS( @REGNAME, "ADC0_CONTROL" ))
    @IF ( @EQS( @FIELDNAME, "CHANNEL000" ))
        @SET(REGVAL , @REGVAL & ( ~ @ADC0_CONTROL_CHANNEL000_MSK ))
    ))) @SET(REGVAL , @REGVAL | (@ADC0_CONTROL_CHANNEL000_MSK & (@VALUE SHL @ADC0_CONTROL_CHANNEL000_POS
        @SET(FOUND,1)
    @ENDI

    @IF ( @EQS( @FIELDNAME, "CHANNEL001" ))
```

```
        @SET(REGVAL , @REGVAL & ( ~ @ADC0_CONTROL_CHANNEL001_MSK ))
    )))@SET(REGVAL , @REGVAL | (@ADC0_CONTROL_CHANNEL001_MSK & (@VALUE SHL @ADC0_CONTROL_CHANNEL001_POS
        @SET(FOUND,1)
    @ENDI

    @IF ( @EQS( @FIELDNAME, "CHANNEL002" ))
        @SET(REGVAL , @REGVAL & ( ~ @ADC0_CONTROL_CHANNEL002_MSK ))
    )))@SET(REGVAL , @REGVAL | (@ADC0_CONTROL_CHANNEL002_MSK & (@VALUE SHL @ADC0_CONTROL_CHANNEL002_POS
        @SET(FOUND,1)
    @ENDI

    @IF ( @EQS( @FIELDNAME, "CHANNEL003" ))
        @SET(REGVAL , @REGVAL & ( ~ @ADC0_CONTROL_CHANNEL003_MSK ))
    )))@SET(REGVAL , @REGVAL | (@ADC0_CONTROL_CHANNEL003_MSK & (@VALUE SHL @ADC0_CONTROL_CHANNEL003_POS
        @SET(FOUND,1)
    @ENDI
    ...
    ...
@ENDI
...
...
@IF( @FOUND == 0 )
@OUT ( "ERROR: NO FIELD AVAILABLE " )
@OUT ( @REGNAME  )
@OUT ( " " )
@OUT ( @FIELDNAME )
@OUT ( "\n" )
@ABORT ( 1 )
@ENDI
@ENDD



; ============================================================================
@*DEFINE SETFIELD(REGNAME,FIELDNAME,SOURCE,TEMP)

@SET(FOUND,0)
@IF ( @EQS( @REGNAME, "ADC0_CONTROL" ))
 @IF ( @EQS ( @FIELDNAME, "CHANNEL000" ) )
     @SET( FOUND , 1)
     PUSH @TEMP
     XOR @TEMP,@TEMP
     SHL @SOURCE,#@ADC0_CONTROL_CHANNEL000_POS
     AND @SOURCE,#@ADC0_CONTROL_CHANNEL000_MSK
     MOV @TEMP,ADC0_CONTROL
     AND @TEMP,#~@ADC0_CONTROL_CHANNEL000_MSK
     OR  @TEMP,@SOURCE
     MOV ADC0_CONTROL,@TEMP
     POP @TEMP
 @ENDI
 ...
 ...
@ENDI
 ...
 ...
```

```
@IF( @FOUND == 0 )
@OUT ( "ERROR: NO FIELD AVAILABLE " )
@OUT ( @REGNAME  )
@OUT ( " " )
@OUT ( @FIELDNAME )
@OUT ( "\n" )
@ABORT ( 1 )
@ENDI
@ENDD



; ============================================================================
; Function    : SETFIELDSV
; Purpose     : Sets the given fields to the constant values specified
; Description : Unspecified fields are set to 0
; ============================================================================

@*DEFINE SETFIELDSV( Parameters )
@MATCH( PAR, List,  @Parameters )
@SET(REGVAL,0)
@MATCH(REGNAME,"")
@MATCH(FLDNAME,"")
@WHILE( @LEN( @PAR ) )
   @IF ( @EQS (@REGNAME, "") )
       @MATCH(REGNAME, @PAR)
   @ELSE
       @MATCH(FLDNAME, @PAR)
       @MATCH(PAR, List, @List )
       @SETFIELDV(@REGNAME, @FLDNAME, @PAR)
   @ENDI
   @MATCH(PAR, List, @List )
@ENDW
MOV    @REGNAME, #@REGVAL
@ENDD

; ============================================================================
@*DEFINE SETREG(REGNAME,SOURCE)
mov @SOURCE,#@REGVAL
mov @REGNAME,@SOURCE
@SET(REGVAL,0)
@ENDD

; ============================================================================
@*DEFINE GETREG(REGNAME,SOURCE)
mov @SOURCE,@REGNAME
@ENDD

; ============================================================================
; End of File ...
```

Assembler Code Example without and with Field Access Layer

```
; Example code using field access layer
;
@SETFIELDV(ADC0_CONTROL,MODE,101b)
@SETFIELDV(ADC0_CONTROL,CHANNEL001,0)
@SETFIELDV(ADC0_CONTROL,ENABLE,1)
@SETREG(ADC0_CONTROL,R3)

; The above lines will be translated exactly to
; the code on the right hand side
; however, if field definitions change,
;the source code here is not affected
```

```
; Example code without using field access layer
;
mov R3,#0x10010100b
mov ADC0_CONTROL,R3



; For not familiar developers this code
; is difficult to understand
; if the specification changed, nobody knows
; what was intended initially. Therefore, comments
; are needed and must be maintained!
; There will be no automatic way to make sure
; the comments are up to date
```

Example TCL-Code used within Regtify's Driver Generator

Used Regtify Version: 1.006

```
# ============================================================================
# Driver    : Tasking Definition File Export Filter
# Author    : TSC
# Revision  : 1.2
# ============================================================================

set regpath "$register::name"
regsub  -all -- "\[{}\]" $regpath "" regpath
regsub  -all -- "\[/\]" $regpath "_" regpath
set accesstype ""
set def 0 ;# Default Value for Complete Register
set rw "" ;# Default Read Write Access

# Calculate Reset Value
for {set i 0 }  {$i < $field::nr}  { incr i} {
    set def   [expr $def + ( $field::default($i) << $field::offset($i) ) ]
}
set directive ""
if  { [expr $register::address      >= 0xc000 && $register::address <= 0xfffe ] } then {
 set directive DEFA ;# Define System-Address
} elseif { [expr $register::address >= 0xfe00 && $register::address <= 0xffde ] } then {
 set directive DEFR ;# Define SFR-Address
} elseif { [expr $register::address >= 0xf000 && $register::address <= 0xf1de] } then {
set directive DEFR ;# Define XSFR-Address
} else {
 set directive "Unknown"
}

fputs   ""
set atnr 0  ;# Access not only read
set atnw 0  ;# Access not only write
set at    "RW"

if { $isPass == 1 && $isFirstRegister } then {
    fputs "; ============================================================================"
    fputs "; File            : $driver::filepath"
    fputs "; Purpose         : Define Macros for Tasking Macro Assembler for Register/Fields  "
    fputs "; System          : $system::name"
    fputs "; Generation Date : [clock format [clock seconds] -format %Y-%m-%d\ %H:%M:%S]"
    fputs "; ============================================================================ "
    fputs ""
}


if { $isPass == 2 && $isFirstRegister } then {
    fputs "; ============================================================================"
    fputs "@*DEFINE SETFIELDV(REGNAME,FIELDNAME,VALUE)"
    fputs ""
    fputs "@SET(FOUND,0)"
}

if { $isPass == 3 && $isFirstRegister } then {
```

```
    fputs "; ================================================================================"
    fputs "@*DEFINE SETFIELD(REGNAME,FIELDNAME,SOURCE,TEMP)"
    fputs ""
    fputs "@SET(FOUND,0)"
}

if { $isPass == 4 && $isFirstRegister } then {
    fputs "; ================================================================================"
    fputs "@*DEFINE GETFIELD(TARGET,REGNAME,FIELDNAME)"
    fputs ""
    fputs "@SET(FOUND,0)"
}


if { $isPass == 2 || $isPass == 3 || $isPass == 4 } then {
    fputs "@IF ( @EQS( @REGNAME, \"${regpath}\" ))"
}



for {set i 0 }  {$i < $field::nr}  { incr i} {
    set fieldname $field::name($i)
    regsub  -all -- "\[{}\]" $fieldname ""  fieldname
    regsub  -all -- "\[/\]"  $fieldname "_" fieldname
    set     fieldpath         "${regpath}_$fieldname"

    switch -regexp -- $field::type($i)  \
        {^[rR][hH]?$}   { set at "R" }  \
        {^[wW][hH]?$}   { set at "W" }

    set fieldposition $field::offset($i)

    # Create field mask
    set fieldmask "b"
    for { set j 0 } { $j < 16 } { incr j } {
      if { $j < $fieldposition || $j > $fieldposition +  $field::size($i) - 1 } then {
        set fieldmask "0$fieldmask"
      } else {
        set fieldmask "1$fieldmask"
      }
    }

    if { $isPass == 1 } then {
     fputs "@SET(${fieldpath}_MSK,$fieldmask)"
     fputs "@SET(${fieldpath}_POS,$fieldposition)"
    }


    if { $isPass == 2 } then {
     fputs "    @IF ( @EQS( @FIELDNAME, \"${fieldname}\" ))"
     fputs "        @SET(REGVAL , @REGVAL & ( ~ @${fieldpath}_MSK ))"
     fputs "        @SET(REGVAL , @REGVAL | (@${fieldpath}_MSK & (@VALUE SHL @${fieldpath}_POS )))"
     fputs "        @SET(FOUND,1)"
     fputs "    @ENDI"
     fputs ""
    }
```

```
    if { $isPass == 3 } then {
        fputs " @IF ( @EQS ( @FIELDNAME, \"$fieldname\" ) )"
        fputs "     @SET( FOUND , 1)"
        fputs "     PUSH @TEMP"
        fputs "     XOR @TEMP,@TEMP"
        fputs "     SHL @SOURCE,#@${fieldpath}_POS"
        fputs "     AND @SOURCE,#@${fieldpath}_MSK"
        fputs "     MOV @TEMP,$regpath"
        fputs "     AND @TEMP,#~@${fieldpath}_MSK"
        fputs "     OR  @TEMP,@SOURCE"
        fputs "     MOV $regpath,@TEMP"
        fputs "     POP @TEMP"
        fputs " @ENDI"
        fputs ""
    }

    if { $isPass == 4 } then {
        fputs " @IF ( @EQS ( @FIELDNAME, \"$fieldname\" ) )"
        fputs "     @SET( FOUND , 1)"
        fputs "     MOV @TARGET,$regpath"
        fputs "     AND @TARGET,#@${fieldpath}_MSK"
        fputs "     SHR @TARGET,#@${fieldpath}_POS"
        fputs " @ENDI"
        fputs ""
    }
}


if { $isPass == 2 || $isPass == 3 || $isPass == 4 } then {
    fputs "@ENDI"
}


if { $isLastRegister && ( $isPass == 2 || $isPass == 3 || $isPass == 4 ) } then {
    fputs "@IF( @FOUND == 0 )"
    fputs "@OUT ( \"ERROR: NO FIELD AVAILABLE \" )"
    fputs "@OUT ( @REGNAME  ) "
    fputs "@OUT ( \" \" )    "
    fputs "@OUT ( @FIELDNAME ) "
    fputs "@OUT ( \"\\n\" ) "
    fputs "@ABORT ( 1 )"
    fputs "@ENDI"
    fputs "@ENDD"
    fputs ""
    fputs ""
}

if { $isLastRegister && $isPass == 3 } then {
    fputs ""
}


if { $isPass == 4 && $isLastRegister } then {
```

```
    fputs "; ============================================================================="
    fputs "; Function    : SETFIELDSV"
    fputs "; Purpose     : Sets the given fields to the constant values specified"
    fputs "; Description : Unspecified fields are set to 0"
    fputs "; ============================================================================="
    fputs ""
    fputs "@*DEFINE SETFIELDSV( Parameters )"
    fputs "@MATCH( PAR, List,  @Parameters )"
    fputs "@SET(REGVAL,0)"
    fputs "@MATCH(REGNAME,\"\")"
    fputs "@MATCH(FLDNAME,\"\")"
    fputs "@WHILE( @LEN( @PAR ) )"
    fputs "   @IF ( @EQS (@REGNAME, \"\") )"
    fputs "       @MATCH(REGNAME, @PAR)"
    fputs "   @ELSE"
    fputs "       @MATCH(FLDNAME, @PAR)"
    fputs "       @MATCH(PAR, List, @List )"
    fputs "       @SETFIELDV(@REGNAME, @FLDNAME, @PAR)"
    fputs "   @ENDI"
    fputs "   @MATCH(PAR, List, @List )"
    fputs "@ENDW"
    fputs "MOV    @REGNAME, #@REGVAL"
    fputs "@ENDD"
    fputs ""

    fputs "; ============================================================================="
    fputs "@*DEFINE SETREG(REGNAME,SOURCE)"
    fputs "mov @REGNAME,@SOURCE"
    fputs "@SET(REGVAL,0)"
    fputs "@ENDD"
    fputs ""

    fputs "; ============================================================================="
    fputs "@*DEFINE GETREG(REGNAME,SOURCE)"
    fputs "mov @SOURCE,@REGNAME"
    fputs "@ENDD"
    fputs ""
    fputs "; ============================================================================="
    fputs "; End of File $driver::filepath"
    fputs ""
    exitDriver
}

; End of Driver Code
```

Outlook: Of course, the same principles deployed herein can be used for different environments and processor platforms, also for the C and C++ languages. **Regtify** generated code can be also used to control debugging and verification sessions.