(\*

\* This file is part of matiec - a compiler for the programming

\* languages defined in IEC 61131-3

\*

\* Copyright (C) 2011 Mario de Sousa (msousa@fe.up.pt)

\*

\* See COPYING and COPYING.LESSER files for copyright details.

\* This library is free software; you can redistribute it and/or

\* modify it under the terms of the GNU Lesser General Public

\* License as published by the Free Software Foundation; either

\* version 3 of the License, or (at your option) any later version.

\*

\* This library is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

\* Lesser General Public License for more details.

\*

\* You should have received a copy of the GNU Lesser General Public

\* License along with this library; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307

\* USA

\*

\* This code is made available on the understanding that it will not be

\* used in safety-critical situations without a full and competent review.

\*)

(\*

\* An IEC 61131-3 compiler.

\*

\* Based on the

\* FINAL DRAFT - IEC 61131-3, 2nd Ed. (2001-12-10)

\*

\*)

(\*

\* This is part of the library conatining the functions

\* and function blocks defined in the standard.

\*

\* Timer Function Blocks

\* ---------------------

\*

\* NOTE: The timing diagrams in the comments (except the state variable)

\* were taken from the IEC 61131-3 standard.

\*)

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TP - pulse timing - state machine

+--------+ ++ ++ +--------+

IN | | || || | |

--+ +-----++-++---+ +---------

t0 t1 t2 t3 t4 t5

+----+ +----+ +----+

Q | | | | | |

--+ +---------+ +--+ +-------------

t0 t0+PT t2 t2+PT t4 t4+PT

PT +---+ + +---+

: / | /| / |

ET : / | / | / |

: / | / | / |

: / | / | / |

0-+ +-----+ +--+ +---------

t0 t1 t2 t4 t5

2 +---+ + +---+

STATE 1 +----+ | +----| +----+ |

0 --+ +-----+ +--+ +---------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

FUNCTION\_BLOCK TP

VAR\_INPUT

IN : BOOL; (\* first input parameter \*)

PT : TIME; (\* second input parameter \*)

END\_VAR

VAR\_OUTPUT

Q : BOOL := FALSE; (\* first output parameter \*)

ET : TIME := T#0s; (\* second output parameter \*)

END\_VAR

VAR

STATE : SINT := 0; (\* internal state: 0-reset, 1-counting, 2-set \*)

PREV\_IN : BOOL := FALSE;

CURRENT\_TIME, START\_TIME : TIME;

END\_VAR

{\_\_SET\_VAR(data\_\_->,CURRENT\_TIME,,\_\_CURRENT\_TIME)}

IF ((STATE = 0) AND NOT(PREV\_IN) AND IN) (\* found rising edge on IN \*)

THEN

(\* start timer... \*)

STATE := 1;

Q := TRUE;

START\_TIME := CURRENT\_TIME;

ELSIF (STATE = 1)

THEN

IF ((START\_TIME + PT) <= CURRENT\_TIME)

THEN

STATE := 2;

Q := FALSE;

ET := PT;

ELSE

ET := CURRENT\_TIME - START\_TIME;

END\_IF;

END\_IF;

IF ((STATE = 2) AND NOT(IN))

THEN

ET := T#0s;

STATE := 0;

END\_IF;

PREV\_IN := IN;

END\_FUNCTION\_BLOCK

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TON - On-delay timing - state machine

+--------+ +---+ +--------+

IN | | | | | |

--+ +--------+ +---+ +-------------

t0 t1 t2 t3 t4 t5

+---+ +---+

Q | | | |

-------+ +---------------------+ +-------------

t0+PT t1 t4+PT t5

PT +---+ +---+

: / | + / |

ET : / | /| / |

: / | / | / |

: / | / | / |

0-+ +--------+ +---+ +-------------

t0 t1 t2 t3 t4 t5

2 +---+ +---+

STATE 1 +----+ | +---+ +----+ |

0 --+ +--------+ +---+ +------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

FUNCTION\_BLOCK TON

VAR\_INPUT

IN : BOOL; (\* first input parameter \*)

PT : TIME; (\* second input parameter \*)

END\_VAR

VAR\_OUTPUT

Q : BOOL := FALSE; (\* first output parameter \*)

ET : TIME := T#0s; (\* second output parameter \*)

END\_VAR

VAR

STATE : SINT := 0; (\* internal state: 0-reset, 1-counting, 2-set \*)

PREV\_IN : BOOL := FALSE;

CURRENT\_TIME, START\_TIME : TIME;

END\_VAR

{\_\_SET\_VAR(data\_\_->,CURRENT\_TIME,,\_\_CURRENT\_TIME)}

IF ((STATE = 0) AND NOT(PREV\_IN) AND IN) (\* found rising edge on IN \*)

THEN

(\* start timer... \*)

STATE := 1;

Q := FALSE;

START\_TIME := CURRENT\_TIME;

ELSE

(\* STATE is 1 or 2 !! \*)

IF (NOT(IN))

THEN

ET := T#0s;

Q := FALSE;

STATE := 0;

ELSIF (STATE = 1)

THEN

IF ((START\_TIME + PT) <= CURRENT\_TIME)

THEN

STATE := 2;

Q := TRUE;

ET := PT;

ELSE

ET := CURRENT\_TIME - START\_TIME;

END\_IF;

END\_IF;

END\_IF;

PREV\_IN := IN;

END\_FUNCTION\_BLOCK

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TOF - Off-delay timing - state machine

+--------+ +---+ +--------+

IN | | | | | |

---+ +--------+ +---+ +-----------

t0 t1 t2 t3 t4 t5

+-------------+ +---------------------+

Q | | | |

---+ +---+ +------

t0 t1+PT t2 t5+PT

PT +---+ +------

: / | + /

ET : / | /| /

: / | / | /

: / | / | /

0------------+ +---+ +--------+

t1 t3 t5

2 +---+ +------

STATE 1 +----+ | +---+ +----+

0 -------------+ +---+ +--------+

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

FUNCTION\_BLOCK TOF

VAR\_INPUT

IN : BOOL; (\* first input parameter \*)

PT : TIME; (\* second input parameter \*)

END\_VAR

VAR\_OUTPUT

Q : BOOL := FALSE; (\* first output parameter \*)

ET : TIME := T#0s; (\* second output parameter \*)

END\_VAR

VAR

STATE : SINT := 0; (\* internal state: 0-reset, 1-counting, 2-set \*)

PREV\_IN : BOOL := FALSE;

CURRENT\_TIME, START\_TIME : TIME;

END\_VAR

{\_\_SET\_VAR(data\_\_->,CURRENT\_TIME,,\_\_CURRENT\_TIME)}

IF ((STATE = 0) AND PREV\_IN AND NOT(IN)) (\* found falling edge on IN \*)

THEN

(\* start timer... \*)

STATE := 1;

START\_TIME := CURRENT\_TIME;

ELSE

(\* STATE is 1 or 2 !! \*)

IF (IN)

THEN

ET := T#0s;

STATE := 0;

ELSIF (STATE = 1)

THEN

IF ((START\_TIME + PT) <= CURRENT\_TIME)

THEN

STATE := 2;

ET := PT;

ELSE

ET := CURRENT\_TIME - START\_TIME;

END\_IF;

END\_IF;

END\_IF;

Q := IN OR (STATE = 1);

PREV\_IN := IN;

END\_FUNCTION\_BLOCK