M851 WristApp API Reference Guide

Timex Corporation
January 26, 2004

DOCUMENT REVISION HISTORY

REVISION: 1.0	DATE: 07/18/2002	Niño Aldrin L. Sarmiento

AFFECTED PAGES	DESCRIPTION
All	Created document.

REVISION: 1.1	DATE: 04/09/2003	Niño Aldrin L. Sarmiento

AFFECTED PAGES	DESCRIPTION
114	Updated example document for API LCD_DISP_SMALL_FIXED_WIDTH_2DIG_DM_DATA_NO_ZERO_SUP

REVISION: 1.2	DATE: 05/27/2003	Niño Aldrin L. Sarmiento

AFFECTED PAGES	DESCRIPTION
229	Updated example document for API KSTP_COPY_RESOURCE_TO_BUFFER

REVISION: 1.3	DATE: 07/10/2003	Niño Aldrin L. Sarmiento

AFFECTED PAGES	DESCRIPTION
117	Remove a line in sample code for the API LCD_DISP_BIG_2DIGIT_DM_DATA_NO_ZERO_SUP
61-73	Updated examples for database APIs
127	Update example for LCD_SET_BLINK_FLAG_STATUS
196-197	Updated example for KTOD_SUBTRACT_DAYS

REVISION: 1.4	DATE: 01/26/2004	Niño Aldrin L. Sarmiento

AFFECTED PAGES	DESCRIPTION
57	Change IY to HL as the base address of new melody. Updated sample code.

TABLE OF CONTENTS

Introduction

This reference guide is divided into two parts:

- "API Summary" provides a quick reference for looking up the API required to handle specific functions.
- "Detailed API Reference" provides the usage of an API along with its required input parameters, assumptions, expected output and microcontroller registers affected by the operation.

NOTE: *The information in this reference guide is the best available at the time of release. This will be updated from time to time.*

Document Convention

This manual uses the following typographic conventions.

Example of Convention	Description
{ `ON` | `OFF` }	In syntax, braces and vertical bar indicate a mandatory choice between two or more items.
{ *expression_list* }	In syntax, items in braces and in italics should be replaced with a constant.

API Summary

CORE API

Changing Foreground Application

```
CORE_REQ_MODE_CHANGE
CORE_REQ_MODE_CHANGE_NEXT
CORE_REQ_MODE_CHANGE_NEXT_NO_PEEK
CORE_FORCE_PRIMARY_AS_NEXT_MODE
```

Changing Application Foreground State Handler

```
CORE_REQ_STATE_CHANGE
CORE_REQ_STATE_CHANGE_NO_CLEAR_DISPLAY
```

Requesting Timeouts

```
CORE_REQ_TIMEOUT_HIRES
CORE_REQ_TIMEOUT_LORES
CORE_REQ_TIMEOUT_STICKY
CORE_CANCEL_TIMEOUTS
```

Peeking At Background Application Data

```
CORE_REQ_PEEK_APP_TYPE
```

Modifying System Operation

- CORE_SET_AUTORETURN
- CORE_CLEAR_AUTORETURN

Blinking and Scrolling Operations

- CORE_REQ_BLINK_2HZ
- CORE_CANCEL_BLINK_2HZ
- CORE_REQ_BLINK_4HZ
- CORE_CANCEL_BLINK_4HZ
- CORE_CANCEL_SCROLLING

Primary Mode Icon Operations

- CORE_BACKGROUND_ICON_REFRESH_ENABLE
- CORE_BACKGROUND_ICON_REFRESH_DISABLE

Switch, Ring and Crown Operations

- CORE_ALLOW_KEYS
- CORE_MASK_KEYS
- CORE_ALLOW_ALL_KEYMASK
- CORE_SUSPEND_RING_EVENTS
- CORE_ENABLE_RING_EVENTS
- CORE_SUSPEND_RING_LEADING_EVENTS
- CORE_ENABLE_RING_LEADING_EVENTS
- CORE_ENABLE_SWITCH_RELEASE
- CORE_SUSPEND_SWITCH_RELEASE
- CORE_ENABLE_GENERIC_SWITCH_EVENT
- CORE_DISABLE_GENERIC_SWITCH_EVENT
- CORE_REQUEST_MELODY_POPUP_CANCEL
- CORE_ENABLE_PULSE_MODE
- CORE_DISABLE_PULSE_MODE
- HW_KBD_CANCEL_CURRENT_SWITCH_RELEASE

Background Task Operations

- CORE_REQ_BACKGROUND_TASK
- CORE_REQ_BACKGROUND_TASK_WITH_PRIORITYCHECK
- CORE_REQ_BACKGROUND_TASK_FOR_APPTYPE
- CORE_REQ_BACKGROUND_TASK_FOR_APPTYPE_WITH_PRIORITYCHECK
- CORE_REQ_BACKGROUND_TASK_FOR_PTZBASEDAPPS
- CORE_CLEAR_BACKGROUND_APPLICATION_TASK
- CORE_CLEAR_ALL_BACKGROUND_APPLICATION_TASK

Popup Operations

- CORE_REQ_POPDOWN
- CORE_SET_POPDOWN_STATE
- CORE_ENABLE_POPUPS
- CORE_SUSPEND_POPUPS
- CORE_CANCEL_ALL_POPUPS
- CORE_CANCEL_ANY_PTZ_POPUPS
- CORETASK_POPUP_REQUEST_BCK
- CORETASK_POPUP_CANCEL_BCK
- CORETASK_POPUP_REQUEST_TZC
- CORETASK_POPUP_CANCEL_TZC
- CORETASK_POPUP_REQUEST_TMR

CORETASK_POPUP_CANCEL_TMR
CORETASK_POPUP_REQUEST_STP
CORETASK_POPUP_CANCEL_STP
CORETASK_POPUP_REQUEST_SYN
CORETASK_POPUP_CANCEL_SYN

Utilities

CORE_ENABLE_SET_MODE
CORE_DISABLE_SET_MODE
CORE_SET_RANDOM_SEED
CORE_GENERATE_RANDOM_NUMBER
CORE_DATABASE_MODIFIED_BY_USER
CORE_APP_ID_PRESENT
CORE_GET_APP_INDEX
CORE_GET_APP_INDEX_FROM_APP_TYPE

AUDIO API

AUDSTART_SYSTEM_MELODY
AUDSETUP_MELODYADDRESS
AUDSTOP_MELODY

DATABASE API

Open and Close Database Operations

DB_OPEN_FILE
DB_OPEN_FILE_LINK_LIST
DB_CLOSE_FILE

Database Information Header Operations

DB_READ_APPLICATION_INFOHEADER
DB_WRITE_APPLICATION_INFOHEADER
DB_GET_ABSOLUTE_ADDRESS_OF_RECORD_RANDOMVAR
DB_GET_ABSOLUTE_ADDRESS_OF_RECORD_RANDOMFIX

Record Read and Write Operations

DB_WRITE_RECORD
DB_WRITE_RECORD_WITHOFFSET
DB_WRITE_RECORD_RANDOMFIX
DB_WRITE_RECORD_WITHOFFSET_RANDOMFIX
DB_WRITE_RECORD_RANDOMVAR
DB_WRITE_RECORD_WITHOFFSET_RANDOMVAR
DB_READ_RECORD
DB_READ_RECORD_WITHOFFSET
DB_READ_RECORD_RANDOMFIX
DB_READ_RECORD_WITHOFFSET_RANDOMFIX
DB_READ_RECORD_RANDOMVAR
DB_READ_RECORD_WITHOFFSET_RANDOMVAR

Linked-List Operations

DB_REMOVE_RECORD_LINKLIST
DB_INSERT_RECORD_LINKLIST
DB_LOCATE_INSERTION_BYSIZE_LINKLIST

DISPLAY API

Flag Display

```
LCD_CLR_ALL_FLAGS
LCD_UPD_L_FLAG
LCD_UPD_A_FLAG
LCD_UPD_P_FLAG
LCD_UPD_LAP_FLAG
LCD_UPD_AP_FLAG
LCD_UPD_NOTE_FLAG
LCD_UPD_ACLK_FLAG
LCD_UPD_TAIL_FLAG
LCD_UPD_ARROW_FLAG
LCD_UPD_TMR_FLAG
LCD_UPD_CDC_FLAG
LCD_UPD_CDR_FLAG
LCD_UPD_CHR_FLAG
LCD_UPD_MOON_FLAG
LCD_UPD_DASH1_FLAG
LCD_UPD_PER1_FLAG
LCD_UPD_COLON1_FLAG
LCD_UPD_DASH2_FLAG
LCD_UPD_PER2_FLAG
LCD_UPD_COLON2_FLAG
```

Clearing Display

```
LCD_CLR_2DIGIT_SEG
LCD_CLR_3DIGIT_SEG
LCD_CLR_SMALL_PROP_WIDTH_2DIG_DM_DATA
LCD_CLR_SMALL_FIXED_WIDTH_2DIG_DM_DATA
LCD_CLR_SMALL_PROP_WIDTH_3DIG_DM_DATA
LCD_CLR_SMALL_FIXED_WIDTH_3DIG_DM_DATA
```

Clearing Large-Font Digits

```
LCD_CLR_BIG_2DIGIT_DM_DATA
LCD_CLR_BIG_3DIGIT_DM_DATA
```

Clearing Display Regions

```
LCD_CLR_DISPLAY
LCD_FILL_DISPLAY
LCD_CLR_SEG_LINE
LCD_CLEAR_UPPER_DM
LCD_CLEAR_MAIN_DM
LCD_CLR_MAIN_DM_LINE1
LCD_CLR_MAIN_DM_LINE2
```

Data Formatting

```
LCD_FORMAT_3DIGIT_DATA_SUP_ZERO
LCD_FORMAT_2DIGIT_DATA_SUP_ZERO
LCD_FORMAT_3DIGIT_DATA_NO_LSD_SUP
LCD_FORMAT_2DIGIT_DATA_SUP_ZERO_MSD
LCD_FORMAT_3DIGIT_DATA_NO_ZERO_SUP
LCD_FORMAT_2DIGIT_DATA_NO_ZERO_SUP
```

Character Display

```
LCD_DISP_SEG_CHAR
LCD_DISP_SMALL_PROP_WIDTH_DM_CHAR
LCD_DISP_SMALL_FIXED_WIDTH_DM_CHAR
```

Large-Font Character Display

```
LCD_DISP_BIG_DM_CHAR
```

Numeric Display

```
LCD_DISP_2DIG_SEG_DATA_WITH_ZERO_SUP
LCD_DISP_3DIG_SEG_DATA_WITH_ZERO_SUP
LCD_DISP_2DIG_SEG_DATA_SUP_ZERO_MSD
LCD_DISP_3DIG_SEG_DATA_NO_LSD_SUP
LCD_DISP_2DIG_SEG_DATA_NO_ZERO_SUP
LCD_DISP_3DIG_SEG_DATA_NO_ZERO_SUP
LCD_DISP_4DIG_SEG_DATA_WITH_ZERO_SUP
LCD_DISP_SMALL_PROP_WIDTH_2DIG_DM_DATA_SUP_ZERO
LCD_DISP_SMALL_FIXED_WIDTH_2DIG_DM_DATA_SUP_ZERO
LCD_DISP_SMALL_PROP_WIDTH_3DIG_DM_DATA_SUP_ZERO
LCD_DISP_SMALL_FIXED_WIDTH_3DIG_DM_DATA_SUP_ZERO
LCD_DISP_SMALL_PROP_WIDTH_2DIG_DM_DATA_SUP_ZERO_MSD
LCD_DISP_SMALL_FIXED_WIDTH_2DIG_DM_DATA_SUP_ZERO_MSD
LCD_DISP_SMALL_PROP_WIDTH_3DIG_DM_DATA_NO_LSD_SUP
LCD_DISP_SMALL_FIXED_WIDTH_3DIG_DM_DATA_NO_LSD_SUP
LCD_DISP_SMALL_PROP_WIDTH_2DIG_DM_DATA_NO_ZERO_SUP
LCD_DISP_SMALL_FIXED_WIDTH_2DIG_DM_DATA_NO_ZERO_SUP
LCD_DISP_SMALL_FIXED_WIDTH_3DIG_DM_DATA_NO_ZERO_SUP
```

Large-Font Numeric Display

```
LCD_DISP_BIG_2DIGIT_DM_DATA_SUP_ZERO
LCD_DISP_BIG_3DIGIT_DM_DATA_SUP_ZERO
LCD_DISP_BIG_2DIGIT_DM_DATA_SUP_ZERO_MSD
LCD_DISP_BIG_3DIGIT_DM_DATA_NO_LSD_SUP
LCD_DISP_BIG_2DIGIT_DM_DATA_NO_ZERO_SUP
LCD_DISP_BIG_3DIGIT_DM_DATA_NO_ZERO_SUP
```

Message Display

```
LCD_DISP_SEG_LINE_MSG
LCD_DISP_FORMATTED_SMALL_PROP_WIDTH_DM_MSG
LCD_DISP_FORMATTED_SMALL_FIXED_WIDTH_DM_MSG
LCD_DISP_UNFORMATTED_SMALL_PROP_WIDTH_DM_MSG
LCD_DISP_UNFORMATTED_SMALL_FIXED_WIDTH_DM_MSG
```

Banner Message Display

```
LCD_DISP_BANNER_MSG
```

Large-Font Message Display

```
LCD_DISP_FORMATTED_BIG_FONT_DM_MSG
LCD_DISP_UNFORMATTED_BIG_FONT_DM_MSG
```

Icon/Flag Blinking

```
LCD_SET_BLINK_FLAG_STATUS
```

LCD_CLR_BLINK_FLAG_STATUS
LCD_CANCEL_ALL_BLINK_FLAGS

Blinking

LCD_WRITE_4HZ_GEN_BLINK_DISP_ROUTINE_ADDR
LCD_WRITE_4HZ_GEN_BLINK_CLR_ROUTINE_ADDR
LCD_WRITE_4HZ_GEN_BLINK_POSITION
LCD_WRITE_4HZ_GEN_BLINK_DISP_ROUTINE_PRELOADED
LCD_WRITE_4HZ_GEN_BLINK_CLR_ROUTINE_PRELOADED
LCD_WRITE_4HZ_GEN_BLINK_POSITION_PRELOADED
LCD_DISP_4HZ_DATA_FIRST

Scrolling

LCD_SCROLL_RAM_OR_ROM_MSG_MAIN_DM_LINE1
LCD_SCROLL_RAM_OR_ROM_MSG_MAIN_DM_LINE2
LCD_PAUSE_SCROLLING
LCD_RESUME_SCROLLING
LCD_SCROLL_MSG_LEFT
LCD_SCROLL_MSG_RIGHT

Primary Mode Icon Resource

LCD_UPDATE_TOD_FLAG_RESOURCE_STATE
LCD_UPDATE_TIMELINE_RESOURCE

Pixel Operations

LCD_DISPLAY_MAIN_DM_PIXEL
LCD_CLEAR_MAIN_DM_PIXEL
LCD_GET_STATE_OF_PIXEL

Display Canned Messages

LCD_DISP_UPPER_DM_MSG_ONE_HALF
LCD_DISP_UPPER_DM_MSG_TO
LCD_DISP_UPPER_DM_MSG_LO
LCD_DISP_SMALL_DM_MSG_HOLD_TO_SWITCH
LCD_DISP_MSG_NO_APPT_UPCOMING
LCD_DISP_SMALL_DM_MSG_NO_OCCASION_UPCOMING
LCD_DISP_SMALL_DM_MSG_SET_TIME_SLASH_DATE
LCD_DISP_SMALL_DM_MSG_SHOW_DAY_OF_WEEK
LCD_DISP_SMALL_DM_MSG_SHOW_WEEK_NUMBER
LCD_DISP_SMALL_DM_MSG_FREE_MEMORY_LAPS
LCD_DISP_SMALL_DM_MSG_TO_STORE_TURN_CROWN
LCD_DISP_SMALL_DM_MSG_MEMORY_FULL
LCD_DISP_SMALL_DM_MSG_HOLD_TO_CLR_ALL
LCD_DISP_SMALL_DM_MSG_HOLD_TO_RESET
LCD_DISP_SMALL_DM_MSG_WORKOUT_STORED
LCD_DISP_SMALL_DM_MSG_RECALL_SLASH_FORMAT
LCD_DISP_SMALL_DM_MSG_FORMAT
LCD_DISP_SMALL_DM_MSG_HOLD_TO_CLEAR_WORKOUT
LCD_DISP_SMALL_DM_MSG_HOLD_TO_CLEAR
LCD_DISP_SMALL_DM_MSG_YOU_ROCK
LCD_DISP_MSG_LAP_SPLIT
LCD_DISP_MSG_SPLIT_LAP
LCD_DISP_MSG_TIME_SPLIT
LCD_DISP_LARGE_DM_STOP
LCD_DISP_LARGE_DM_TOTAL
LCD_DISP_MSG_TIME_LAP

```
LCD_DISP_SMALL_DM_MSG_SET_TIMER
LCD_DISP_SMALL_DM_MSG_UNUSED_ENTRY
LCD_DISP_SMALL_DM_MSG_UNUSED_ENTRIES
LCD_DISP_SMALL_DM_MSG_STOP_AT_END
LCD_DISP_SMALL_DM_MSG_REPEAT_AT_END
LCD_DISP_SMALL_DM_MSG_CHRONO_AT_END
LCD_DISP_SMALL_DM_MSG_SET_FORMAT
LCD_DISP_SMALL_DM_MSG_SET_COUNTER
LCD_DISP_SMALL_DM_MSG_COUNT_UP
LCD_DISP_SMALL_DM_MSG_COUNT_DOWN
LCD_DISP_SMALL_DM_MSG_HOLD_TO_DELETE
LCD_DISP_SMALL_DM_MSG_SET_ALARM
LCD_DISP_SMALL_DM_MSG_ENTRY_DELETED
LCD_DISP_SMALL_DM_MSG_SET_APPT
LCD_DISP_SMALL_DM_MSG_APPT_DATE
LCD_DISP_SMALL_DM_MSG_1ST_APPT_DATE
LCD_DISP_SMALL_DM_MSG_EDIT_NOTE
LCD_DISP_SMALL_DM_MSG_BUTTON_BEEP_ON
LCD_DISP_SMALL_DM_MSG_BUTTON_BEEP_OFF
LCD_DISP_SMALL_DM_MSG_NIGHTMODE_OFF
LCD_DISP_SMALL_DM_MSG_NIGHTMODE_ON
LCD_DISP_SMALL_DM_MSG_NIGHTMODE_AUTO
LCD_DISP_SMALL_DM_MSG_CHIME_OFF
LCD_DISP_SMALL_DM_MSG_CHIME_ON
LCD_DISP_SMALL_DM_MSG_CHIME_AUTO
LCD_DISP_SMALL_DM_MSG_ON_OFF_TIME
LCD_DISP_SMALL_DM_MSG_PASSWORD_NEEDED
LCD_DISP_SMALL_DM_MSG_SORTING_TRY_LATER
LCD_DISP_SMALL_DM_MSG_ENTER_PASSWORD
LCD_DISP_SMALL_DM_MSG_PASSWORD_INVALID
LCD_DISP_SMALL_DM_MSG_PASSWORD
LCD_DISP_SMALL_DM_MSG_NO_ENTRY_SELECTED
LCD_DISP_SMALL_DM_MSG_SNOOZE
LCD_DISP_SMALL_DM_MSG_END_OF_LIST
LCD_DISP_SMALL_DM_MSG_EDIT
LCD_DISP_SMALL_DM_MSG_ON_TIME
LCD_DISP_SMALL_DM_MSG_OFF_TIME
LCD_DISP_SMALL_DM_MSG_COMM_READY
LCD_DISP_SMALL_DM_MSG_COMM_ERROR
LCD_DISP_SMALL_DM_MSG_PUSH_CROWN_IN
LCD_DISP_SEG_MSG_12HR
LCD_DISP_SEG_MSG_24HR
LCD_DISP_SEG_MSG_FREE
LCD_DISP_SEG_MSG_CHRONO
LCD_DISP_SEG_MSG_INT
LCD_DISP_SEG_MSG_HR_MIN
LCD_DISP_SEG_MSG_SECOND
LCD_DISP_SEG_MSG_TOTAL
LCD_DISP_SEG_MSG_DAILY
LCD_DISP_SEG_MSG_WKDAYS
LCD_DISP_SEG_MSG_WKENDS
LCD_DISP_SEG_MSG_WEEKLY
LCD_DISP_SEG_MSG_MNTHLY
LCD_DISP_SEG_MSG_YEARLY
LCD_DISP_SEG_MSG_ALARM
LCD_DISP_SEG_MSG_1_DAY
```

LCD_DISP_SEG_MSG_MMDDYY
LCD_DISP_SEG_MSG_DDMMYY
LCD_DISP_SEG_MSG_YYMMDD
LCD_DISP_SEG_MSG_BATT
LCD_DISP_SEG_MSG_HOLD
LCD_DISP_SEG_MSG_STORE
LCD_DISP_SEG_MSG_TIME
LCD_DISP_SEG_MSG_1_MIN
LCD_DISP_SEG_MSG_ALERT
LCD_DISP_SEG_MSG_SHOW
LCD_DISP_SEG_MSG_COUNT
LCD_DISP_SEG_MSG_SET
LCD_DISP_SEG_MSG_STOP
LCD_DISP_MAIN_DM_LINE1_SELECT
LCD_DISP_MAIN_DM_LINE1_ALARM
LCD_DISP_MAIN_DM_LINE2_ALARM
LCD_DISP_MAIN_DM_LINE1_ALARM_AT
LCD_DISP_MAIN_DM_LINE2_TZ
LCD_DISP_MAIN_DM_LINE2_LAP
LCD_DISP_MAIN_DM_LINE2_LAPS
LCD_DISP_MAIN_DM_LINE1_MEMORY
LCD_DISP_MAIN_DM_LINE2_BEST_LAP
LCD_DISP_MAIN_DM_LINE2_LAP_AVG
LCD_DISP_MAIN_DM_LINE1_APPT_AT
LCD_DISP_MAIN_DM_LINE1_HOURS
LCD_DISP_MAIN_DM_LINE1_MINS
LCD_DISP_MAIN_DM_LINE2_PRIOR
LCD_DISP_MAIN_DM_LINE2_MINS
LCD_DISP_MAIN_DM_LINE1_BDAY
LCD_DISP_MAIN_DM_LINE1_ANNV
LCD_DISP_MAIN_DM_LINE1_HOLIDAY
LCD_DISP_MAIN_DM_LINE1_VACATION
LCD_DISP_MAIN_DM_LINE1_4_DASHES
LCD_DISP_MAIN_DM_LINE1_CHIME
LCD_DISP_MAIN_DM_LINE2_ON
LCD_DISP_MAIN_DM_LINE2_OFF
LCD_DISP_MAIN_DM_LINE2_AUTO
LCD_DISP_MAIN_DM_LINE1_YEAR
LCD_DISP_MAIN_DM_LINE1_AM
LCD_DISP_MAIN_DM_LINE1_PM
LCD_DISP_MAIN_DM_LINE2_AM
LCD_DISP_MAIN_DM_LINE2_PM

Utilities

UTL_DISPLAY_DAY_OF_WEEK
UTL_DISPLAY_DATE_COMPLETE_AND_DOW
UTL_DISPLAY_DATE_COMPLETE
UTL_DISPLAY_HR_MIN_DATA_L1
UTL_DISPLAY_HR_MIN_DATA_L2

TIME ZONE RESOURCE API

Flag Manipulation

KTOD_ACTIVATE_RESOURCE
KTOD_DEACTIVATE_RESOURCE

KTOD_ENABLE_DISP_UPD_SEC_EVENT
KTOD_DISABLE_DISP_UPD_SEC_EVENT
KTOD_ENABLE_DISP_UPD_MIN_EVENT
KTOD_DISABLE_DISP_UPD_MIN_EVENT
KTOD_DEACTIVATE_ALL_DISPLAY_UPDATES
KTOD_MAKE_AS_PRIMARY_TZ
KTOD_SET_FOR_EURO_FORMAT
KTOD_SET_FOR_US_FORMAT

Data Manipulation

KTOD_CALC_DOW
KTOD_CALC_WEEK_NUMBER
KTOD_ADD_DAYS
KTOD_SUBTRACT_DAYS
KTOD_ADD_YEARS
KTOD_SUBTRACT_YEARS
KTOD_CORRECT_MDY_DATA
KTOD_ADJUST_DATE_AND_YEAR

Data Transfer

KTOD_COPY_RESOURCE_TO_RESOURCE
KTOD_COPY_TIME_MIN_TO_DOW_FROM_RESOURCE
KTOD_COPY_TIME_FROM_RESOURCE
KTOD_COPY_MDY_FROM_RESOURCE
KTOD_COPY_HMS_FROM_RESOURCE
KTOD_WRITE_TIME_TO_RESOURCE
KTOD_WRITE_MDY_TO_RESOURCE
KTOD_WRITE_HMS_TO_RESOURCE

Resource Utilities

KTOD_LOAD_RESOURCE_START_ADDRESS
KTOD_GET_UPDATE_FLAGS
KTOD_CHECK_IF_LEAP_YEAR
KTOD_GET_MAX_DAYS_OF_A_MONTH
KTOD_DIVIDE_BY_2
KTOD_CONVERT_BCD_TO_HEX
KTOD_CHECK_IF_DIVISIBLE_BY_4
KTOD_GET_PTZ_ADDRESS
KTOD_GET_TZ_ADDRESS

TIME ZONE CHECK RESOURCE API

Flag Manipulation

KTZC_DEACTIVATE_ALL_RESOURCES
KTZC_ACTIVATE_RESOURCE
KTZC_DISABLE_RESOURCE
KTZC_SETUP_POPUP_GENERATION
KTZC_SETUP_EVENT_GENERATION
KTZC_CANCEL_POPUPEVENT_GENERATION

Data Manipulation

KTZC_RETURN_RESOURCE_OWNER
KTZC_SETUP_MDY_CHECK
KTZC_SETUP_HOUR_MINUTE_MDY

KTZC_SETUP_HOUR_MINUTE
KTZC_SETUP_MONTH
KTZC_SETUP_DATE
KTZC_SETUP_YEAR

BACKUP RESOURCE API

Flag Manipulation

KBCK_DEACTIVATE_ALL_RESOURCES
KBCK_ACTIVATE_RESOURCE
KBCK_DEACTIVATE_RESOURCE
KBCK_SETUP_POPUP_GENERATION
KBCK_SETUP_EVENT_GENERATION
KBCK_CANCEL_POPUPEVENT_GENERATION

Data Manipulation

KBCK_RETURN_RESOURCE_OWNER
KBCK_SETUP_SNOOZE_TIME

STOPWATCH RESOURCE API

Data Manipulation

KSTP_LINK_NEXT_RESOURCE
KSTP_ENABLE_DISP_UPD_EVENT
KSTP_DISABLE_DISP_UPD_EVENT
KSTP_DEACTIVATE_ALL_DISPLAY_UPDATES
KSTP_SETUP_POPUP_GENERATION
KSTP_SETUP_EVENT_GENERATION
KSTP_CANCEL_POPUPEVENT_GENERATION
KSTP_START_RESOURCE
KSTP_STOP_RESOURCE
KSTP_STOP_ALL_RESOURCES
KSTP_TAKE_SPLIT
KSTP_RESET_RESOURCE
KSTP_RESET_ALL_RESOURCE
KSTP_CLEAR_RESOURCE_DATA

Data Transfer

KSTP_COPY_RESOURCE_TO_BUFFER

Resource Utilities

KSTP_COMPARE_BUFFER_FROM_RESOURCE
KSTP_GET_RESOURCE_STATUS

TIMER RESOURCE API

Data Manipulation

KTMR_SETUP_USER_AND_WORK_HMS
KTMR_SETUP_WORKING_HMS
KTMR_SETUP_PREWARNING_HMS
KTMR_DISABLE_PREWARNING_TRACKING
KTMR_ENABLE_DISP_UPD_EVENT

KTMR_DISABLE_DISP_UPD_EVENT
KTMR_DEACTIVATE_ALL_DISPLAY_UPDATES
KTMR_SETUP_COUNTDOWN
KTMR_SET_RUNNING_REPEAT_COUNTER
KTMR_GET_RUNNING_REPEAT_COUNTER
KTMR_SETUP_POPUP_GENERATION
KTMR_SETUP_EVENT_GENERATION
KTMR_CANCEL_POPUP_AND_EVENT_GENERATION
KTMR_SETUP_REPEAT_AT_END
KTMR_CANCEL_REPEAT_AT_END
KTMR_SETUP_TMR_LINK
KTMR_SETUP_STP_LINK
KTMR_CANCEL_ALL_LINKS
KTMR_RESET_RESOURCE
KTMR_START_RESOURCE
KTMR_STOP_RESOURCE

Data Transfer

KTMR_COPY_RESOURCE_TO_BUFFER

Resource Utilities

KTMR_GET_STATUS
KTMR_GET_UPDATE_BITS

SYNCHRO RESOURCE API

Data Manipulation

KSYN_GET_CAUSED_TO_START
KSYN_GET_TOTAL_TIME_UPD_BITS
KSYN_GET_STOPPAGE_TIME_UPD_BITS
KSYN_GET_TOTAL_STATUS
KSYN_GET_STOPPAGE_STATUS
KSYN_ENABLE_DISP_UPD_EVENT
KSYN_DISABLE_DISP_UPD_EVENT
KSYN_SETUP_POPUP_GENERATION
KSYN_SETUP_EVENT_GENERATION
KSYN_CANCEL_POPUPEVENT_GENERATION
KSYN_RESET_AND_START
KSYN_STOP_TOTAL_AND_STOPPAGE_TIME
KSYN_RESET_RESOURCE

Data Transfer

KSYN_COPY_TOTAL_TO_BUFFER
KSYN_COPY_STOPPAGE_TO_BUFFER

UTILITIES API

Generic

KRES_CLEAR_N_BYTES
HW_RESET_MCU
HW_RESET_WATCHDOG
HW_LAMP_POPUP_REQUEST
HW_LAMP_POPUP_REQUEST_OFF

UTL_BINARY_MATH_MODE
UTL_DECIMAL_MATH_MODE

Conversion

UTL_CONVERT_HEX_TO_3DIGIT_BCD
UTL_CONVERT_HEX_TO_2DIGIT_BCD
UTL_CONVERT_2BYTE_HEX_TO_2BYTE_BCD
UTL_CONVERT_1BYTE_BCD_TO_1BYTE_HEX
UTL_CONVERT_4BYTE_FROM_BCD_TO_HEX
UTL_CONVERT_4BYTE_FROM_HEX_TO_BCD
UTL_CONVERT_TO_12HR_FORMAT

Math Functions

UTL_DIVIDE_HMSH_BY_N_IN_HEX
UTL_DIVIDE_HMSH_BY_N_IN_BCD
UTL_ADD_HMSH
UTL_SUBTRACT_HMSH

Copy

UTL_COPY_BUFFER
UTL_COPY_IYREG_TO_IXREG

Comparison

UTL_COMPARE_4BYTE_BUFFER
UTL_COMPARE_HLREG_WITH_IXREG
UTL_COMPARE_HLREG_WITH_IYREG

Acceleration

UTLACCELERATION_1SEC
UTL_ACCELERATION_1INCREMENT
UTLACCELERATION_1MIN
UTLACCELERATION_5MIN
UTL_ACCELERATION_5INCREMENT
UTLACCELERATION_DATE
UTLACCELERATION_DATE

Message Editing

UTL_SETUP_VARS_FOR_EDITING
UTL_FILL_SCROLL_BUFFER_WITH_SPACE
UTL_MOVE_CURSOR_FORWARD_AND_REQ_BLINK
UTL_MOVE_CURSOR_BACKWARD_AND_REQ_BLINK
UTL_DISPLAY_MSG_AND_REQ_BLINK
UTL_POINT_TO_PREV_CHAR_AND_REQ_BLINK
UTL_POINT_TO_NEXT_CHAR_AND_REQ_BLINK
UTL_POINT_TO_PREV_CHAR
UTL_POINT_TO_NEXT_CHAR
UTL_CLEANUP_EDIT_BUFFER
UTL_TRIM_SENTINEL_CHAR
UTL_CHANGE_CHAR_TO_SPACE

Time Structure Math

KRES_CLEARDATABUFFERS
KRES_HUNDSECMINHR_UPD
KRES_SECMINHR_UPD

```
KRES_MINHR_UPD
KRES_HUNDSECMINHR_SUB
KRES_SECMINHR_SUB
KRES_MINHR_SUB
```

API Reference

CORE API

Changing Foreground Application

CORE_REQ_MODE_CHANGE

Description Requests a mode change to a specified mode. If the requested mode is not active, then the system will find the next active mode.

Usage CORE_REQ_MODE_CHANGE

Assumptions None

Input A - Application Index to proceed

Output None

Destroys B

Example

```
; proceed to application 3rd application in the mode list index
ld A, #03H
CORE_REQ_MODE_CHANGE
```

CORE_REQ_MODE_CHANGE_NEXT

Description Requests a mode change to the next mode with primary mode peek operation.

If application is active for more than a specified time, then it will initiate a mode change to the PRIMARY mode. This will enable a system peek to primary mode. If the user, releases the mode switch depression within a specified time period, the system will cancel the mode change request.

Usage CORE_REQ_MODE_CHANGE_NEXT

Assumptions None

Input None

Output None

Destroys B

Example

```
;request a mode change to the next application or TOD
CORE_REQ_MODE_CHANGE_NEXT
```

CORE_REQ_MODE_CHANGE_NEXT_NO_PEEK

Description Requests a mode change to the next mode without a peek to the primary mode.

If application is active for more than a specified time, then it will initiate a mode change to the PRIMARY mode.

Usage CORE_REQ_MODE_CHANGE_NEXT_NO_PEEK

Assumptions None

Input None

Output None

Destroys B

Example
```
;request a mode change to the next application or TOD with no TOD peek
CORE_REQ_MODE_CHANGE_NEXT_NO_PEEK
```

CORE_FORCE_PRIMARY_AS_NEXT_MODE

Description Forces the primary mode to be the next mode on future mode change next requests. This API does not initiate a mode change to the next mode.

Usage CORE_FORCE_PRIMARY_AS_NEXT_MODE

Assumptions None

Input None

Output None

Destroys HL

Example
```
;make TOD the next mode on the next mode change request
CORE_FORCE_PRIMARY_AS_NEXT_MODE
CORE_REQ_MODE_CHANGE_NEXT
```

Changing Application Foreground State Handler

CORE_REQ_STATE_CHANGE

Description Request for a state change. This will clear the entire display area.

Usage CORE_REQ_STATE_CHANGE

Assumptions Required parameters are already setup prior to calling this macro. No checking is done to verify a valid macro input.

Input B - State within the current mode to proceed.

Output None

Destroys B

Example
```
;change state to the set banner state
ld  B, #CORESET1BANNERSTATE
CORE_REQ_STATE_CHANGE
```

CORE_REQ_STATE_CHANGE_NO_CLEAR_DISPLAY

Description Request for a state change without clearing the display area.

Usage CORE_REQ_STATE_CHANGE_NO_CLEAR_DISPLAY

Assumptions Required parameters are already setup prior to calling this macro. No checking is done to verify a valid macro input.

Input B - State within the current mode to proceed.

Output None

Destroys B

Example
```
;change state to the set banner state
ld  B, #CORESET1BANNERSTATE
CORE_REQ_STATE_CHANGE_NO_CLEAR_DISPLAY
```

Requesting Timeouts

CORE_REQ_TIMEOUT_HIRES

Description Request for a high resolution timeout.

Usage CORE_REQ_TIMEOUT_HIRES {*Timeout_Value*}

Assumptions None

Input *Timeout_Value* - Duration before the system event COREEVENT_TIMEOUTDONE_HIGHRES is passed to the foreground application. This value is specified in 125ms increments.

The kernel provides some built in constants for *Timeout_Value*:

Constant	Description
TIMEOUTHIRES_1SEC	1 second timeout
TIMEOUTHIRES_1P5SEC	1.5 second timeout
TIMEOUTHIRES_2SEC	2 second timeout
TIMEOUTHIRES_3SEC	3 second timeout
TIMEOUTHIRES_4SEC	4 second timeout
TIMEOUTHIRES_5SEC	5 second timeout

Output None

Destroys B, A

Example
```
;request a 1.5 second timeout for banner display
CORE_REQ_TIMEOUT_HIRES TIMEOUTHIRES_1P5SEC
```

CORE_REQ_TIMEOUT_LORES

Description Request for a Low resolution timeout.

Usage CORE_REQ_TIMEOUT_LORES {*Timeout_Value*}

Assumptions None

Input *Timeout_Value* - Duration before the system event COREEVENT_TIMEOUTDONE_LOWRES is passed to the foreground application. This value is specified in 1 second increments.

The kernel provides some built in constants for *Timeout_Value*:

Constant	Description
TIMEOUTLORES_2SEC	2 second timeout
TIMEOUTLORES_3SEC	3 second timeout
TIMEOUTLORES_4SEC	4 second timeout
TIMEOUTLORES_10SEC	10 second timeout
TIMEOUTLORES_20SEC	20 second timeout

Output None

Destroys B, A

Example
```
;request a 10 second timeout
CORE_REQ_TIMEOUT_LORES TIMEOUTLORES_10SEC
```

CORE_REQ_TIMEOUT_STICKY

Description Request for a sticky timeout.

Usage CORE_REQ_TIMEOUT_STICKY {*Timeout_Value*}

Assumptions None

Input *Timeout_Value* - Duration before the system event COREEVENT_STICKY_TIMEOUTDONE is passed to the foreground application. This value is specified in 125ms increments.

The kernel provides some built in constants for *Timeout_Value*:

Constant	Description
TIMEOUTHIRES_1SEC	1 second timeout
TIMEOUTHIRES_1P5SEC	1.5 second timeout
TIMEOUTHIRES_2SEC	2 second timeout
TIMEOUTHIRES_3SEC	3 second timeout
TIMEOUTHIRES_4SEC	4 second timeout
TIMEOUTHIRES_5SEC	5 second timeout

Output None

Destroys B, A

Example

```
    ld  A, [CORECurrentEvent]
    cp  A, #COREEVENT_STARTSTOPDEPRESS
    jr  NZ, check_other_events

    ;request a 4 second sticky timeout if STARTSTOP depress detected
    CORE_REQ_TIMEOUT_STICKY TIMEOUTHIRES_4SEC
    ret

check_other_events:

    ld  A, [CORECurrentEvent]
    cp  A, #COREEVENT_STARTSTOPRELEASE
    jr  Z, ProcessReleaseEventIfSwitchReleaseAfterTimeout

    cp  A, #COREEVENT_STICKY_TIMEOUTDONE
    jr  Z, ProcessTimeoutDoneIfSwitchReleaseWithinTimeout
```

CORE_CANCEL_TIMEOUTS

Description Cancel all active timeouts in the system.

Usage CORE_CANCEL_TIMEOUTS

Assumptions This is the default condition whenever a mode or state change occurs.

Input None

Output None

Destroys HL

Example
```
;cancel all active timeout in the system
CORE_CANCEL_TIMEOUTS
```

Peeking At Background Application Data

CORE_REQ_PEEK_APP_TYPE

Description Request the system to search for a specified application type and execute that application's background handler with the background event COREEVENT_PEEK.

Usage CORE_REQ_PEEK_APP_TYPE

Assumptions The application called to be “peek” must support the system event COREEVENT_PEEK in its background handler. If not supported by the application, the display will just be cleared.

If multiple active applications have the same type, then the criteria will be based on mode list order.

Input None
B - Application Type

The kernel provides some built-in constants for application type:

Constant
COREAPPTYPESYSTEM
COREAPPTYPECOMM
COREAPPTYPEOPTION
COREAPPTYPETOD
COREAPPTYPEDATE
COREAPPTYPECHRONO
COREAPPTYPETIMER
COREAPPTYPESYNCHROTIMER
COREAPPTYPECOUNTER
COREAPPTYPECONTACT
COREAPPTYPETASK
COREAPPTYPENOTES
COREAPPTYPESCHEDULE
COREAPPTYPETIDE
COREAPPTYPEDEMO
COREAPPTYPEGAME
COREAPPTYPEALARM
COREAPPTYPEAPPOINTMENT
COREAPPTYPEOCCASION

Output None

Destroys B

Example
```
; peek at upcoming appointments
; this should be called only when there is an appointment app loaded
ld  B, #COREAPPTYPEAPPOINTMENT
CORE_REQ_PEEK_APP_TYPE
```

Modifying System Operation

CORE_SET_AUTORETURN

Description Request for an Auto-Return to Primary mode after 3 minutes of idle activity.

Usage CORE_SET_AUTORETURN

Assumptions None

Input None

Output None

Destroys B

Example
```
;request autoreturn
CORE_SET_AUTORETURN
```

CORE_CLEAR_AUTORETURN

Description Cancel any pending request for Auto-Return to primary mode on idle activity.

Usage CORE_CLEAR_AUTORETURN

Assumptions This is the default condition whenever a mode or state change occurs.

Input None

Output None

Destroys B

Example
```
;do not initiate an autoreturn sequence in this mode
CORE_CLEAR_AUTORETURN
```

Blinking and Scrolling Operations

CORE_REQ_BLINK_2HZ

Description Request blinking of an LCD icon. Blink rate is 2Hz.

Usage CORE_REQ_BLINK_2HZ {*lcd_icon*}

Assumptions None

Input *lcd_icon* - LCD icon to blink

The following constants are available for *lcd_icon*:

Constant	Description
CHR_FLAG	Stopwatch icon
MOON_FLAG	Moon icon
ACLK_FLAG	Alarm Clock icon
ARROW_FLAG	Arrow icon
TAIL_FLAG	Tail icon
TMR_FLAG	Hourglass icon
NOTE_FLAG	Note icon
P_FLAG	P icon
A_FLAG	A icon
L_FLAG	L icon

NOTE: Multiple LCD icons can be ORed together into a single API call.

Output None

Destroys A, B

Example
```
;blink the stopwatch icon
CORE_REQ_BLINK_2HZ CHR_FLAG
```

CORE_CANCEL_BLINK_2HZ

Description Cancel blinking of an LCD icon. Blink rate is 2Hz.

Usage CORE_CANCEL_BLINK_2HZ {*lcd_icon*}

Assumptions None

Input *lcd_icon* - LCD icon to blink

The following constants are available for *lcd_icon*:

Constant	Description
CHR_FLAG	Stopwatch icon
MOON_FLAG	Moon icon
ACLK_FLAG	Alarm Clock icon
ARROW_FLAG	Arrow icon
TAIL_FLAG	Tail icon
TMR_FLAG	Hourglass icon
NOTE_FLAG	Note icon
P_FLAG	P icon
A_FLAG	A icon
L_FLAG	L icon

NOTE: Multiple LCD icons can be ORed together into a single API call.

Output None

Destroys A, B

Example
```
;cancel blinking of the stopwatch icon
CORE_CANCEL_BLINK_2HZ CHR_FLAG
```

CORE_REQ_BLINK_4HZ

Description Request Blinking at 4Hz. The LCD generic 4Hz blinking arguments have been set before calling this macro.

Usage CORE_REQ_BLINK_4HZ

Assumptions None

Input None

Output None

Destroys A, B

Example
```
; setup the display and clear routines for the generic blink engine
LCD_WRITE_4HZ_GEN_BLINK_DISP_ROUTINE_ADDR apptDisplaySetPositionOnly
LCD_WRITE_4HZ_GEN_BLINK_CLR_ROUTINE_ADDR apptClearSetPositionOnly
CORE_REQ_BLINK_4HZ

; force the data to be displayed first
LCD_DISP_4HZ_DATA_FIRST

. . .

apptDisplaySetPositionOnly:
  // put code here to display the data for the generic blink engine
    . . .
```

```
apptClearSetPositionOnly:
   // put code here to clear the display area used in the display
   // display routine
```

CORE_CANCEL_BLINK_4HZ

Description Cancel generic 4Hz blinking

Usage CORE_CANCEL_BLINK_4HZ

Assumptions None

Input None

Output None

Destroys A, B

Example

```
; cancel active generic blinking
CORE_CANCEL_BLINK_4HZ
```

CORE_CANCEL_SCROLLING

Description Cancel the active scrolling operation.

Usage CORE_CANCEL_SCROLLING

Assumptions This is the default condition whenever a mode or state change occurs.

Input None

Output None

Destroys A, B

Example

```
;cancel scrolling
CORE_CANCEL_SCROLLING
```

Primary Mode Icon Operations

CORE_BACKGROUND_ICON_REFRESH_ENABLE

Description Requests the system to inform the foreground application of any updates to the LCD Icon resource.

When the system detects a background operation changing the state of any of the primary mode LCD icon resource, it will send the event `COREEVENT_ICON_REFRESH` to the foreground application for processing.

Usage CORE_BACKGROUND_ICON_REFRESH_ENABLE

Assumptions None

Input None

Output None

Destroys HL

Example

```
;request an event to be passed to the foreground application
;if any of the primary mode lcd resource was updated (foreground or
;background)
CORE_BACKGROUND_ICON_REFRESH_ENABLE
```

CORE_BACKGROUND_ICON_REFRESH_DISABLE

Description Cancels requests informing the foreground application of any updates to the LCD Icon resource.

Usage CORE_BACKGROUND_ICON_REFRESH_ENABLE

Assumptions This is the default condition whenever a mode or state change occurs.

Input None

Output None

Destroys HL

Example

```
;do not pass an event to be passed to the foreground application
;if any of the primary mode lcd resource was updated (foreground or
;background)
CORE_BACKGROUND_ICON_REFRESH_DISABLE
```

Switch, Ring and Crown Operations

CORE_ALLOW_KEYS

Description Allow specified switch events to be passed to the foreground application.

Usage CORE_ALLOW_KEYS {*switch_mask*}

Assumptions None

Input *switch_mask* - Switch bit mask. Switches specified in this mask will be processed by the system.

The following constants are available for *switch_mask*:

Constant	Description
bCOREModeSwitch	MODE switch
bCOREStopResetSwitch	STOP/RESET switch
bCOREStartSplitSwitch	START/SPLIT switch
bCORECWSwitch	RING CW switch
bCORECCWSwitch	RING CCW switch
bCOREELSwitch	EL switch

NOTE:

- EL switch should normally be use only for activating the INDIGLO Night-Light.
- The mask bCOREModeSwitch must be specified for default state handlers to allow mode changes.

Output None

Destroys B

Example

```
;allow only the mode switch to be passed to the state handler
CORE_ALLOW_KEYS bCOREModeSwitch
```

CORE_MASK_KEYS

Description Prevent specified switch events from being passed to the foreground application.

Usage CORE_MASK_KEYS {*switch_mask*}

Assumptions None

Input *switch_mask* - Switch bit mask. Switches specified in this mask will be discarded by the system.

The following constants are available for *switch_mask*:

Constant	Description
bCOREModeSwitch	MODE switch

bCOREStopResetSwitch	STOP/RESET switch
bCOREStartSplitSwitch	START/SPLIT switch
bCORECWSwitch	RING CW switch
bCORECCWSwitch	RING CCW switch
bCOREELSwitch	EL switch

NOTE: EL switch should normally be use only for activating the INDIGLO Night-Light.

Output None

Destroys B

Example
```
;do not pass the ring events to the state handler
CORE_MASK_KEYS (bCORECWSwitch|bCORECCWSwitch)
```

CORE_ALLOW_ALL_KEYMASK

Description Allow all switch events to be passed to the foreground application.

Usage CORE_ALLOW_ALL_KEYMASK

Assumptions None

Input None

Output None

Destroys B

Example
```
;allow all switches to be passed to the foreground application
CORE_ALLOW_ALL_KEYMASK
```

CORE_SUSPEND_RING_EVENTS

Description Suspends any ring events to be passed to the application.

Usage CORE_SUSPEND_RING_EVENTS

Assumptions This is the default operation during a state or mode change

Input None

Output None

Destroys HL

Example
```
;suspend ring events
CORE_SUSPEND_RING_EVENTS
```

CORE_ENABLE_RING_EVENTS

Description Enable any ring events to be passed to the application.

Usage CORE_ENABLE_RING_EVENTS

Assumptions Any of the following switches – `bCORECWSwitch, bCORECCWSwitch` must be allowed in the system. Refer to the CORE_ALLOW_KEYS for activating these switches.

Input None

Output None

Destroys HL

Example
```
;enable ring events to be passed to the foreground application
CORE_ENABLE_RING_EVENTS
```

CORE_SUSPEND_RING_LEADING_EVENTS

Description Suspends any ring events matching CW/CCW leading transitions to be passed to the application.

Usage CORE_SUSPEND_RING_LEADING_EVENTS

Assumptions Any of the following switches – `bCORECWSwitch, bCORECCWSwitch` must be allowed in the system. Refer to the CORE_ALLOW_KEYS for activating these switches.

This is the default condition whenever a mode or state change occurs.

Input None

Output None

Destroys HL

Example
```
;do not allow ring leading events to be passed to the foreground application
CORE_SUSPEND_RING_LEADING_EVENTS
```

CORE_ENABLE_RING_LEADING_EVENTS

Description Enables any ring events matching CW/CCW leading transitions to be passed to the application.

Usage CORE_ENABLE_RING_LEADING_EVENTS

Assumptions Any of the following switches – `bCORECWSwitch, bCORECCWSwitch` must be allowed in the system. Refer to the CORE_ALLOW_KEYS for activating these switches.

Input None

Output None

Destroys HL

Example
```
;allow ring leading events to be passed to the foreground application
CORE_ENABLE_RING_LEADING_EVENTS
```

CORE_ENABLE_SWITCH_RELEASE

Description Enables any switch releases to be passed to the application.

Usage CORE_ENABLE_SWITCH_RELEASE

Assumptions Any of the following switches -- `bCOREModeSwitch, bCOREStopResetSwitch, bCOREStartSplitSwitch` -- must be allowed in the system. Refer to the CORE_ALLOW_KEYS for activating these switches.

Input None

Output None

Destroys HL

Example
```
;allow switch release events to be passed to the foreground application
CORE_ENABLE_SWITCH_RELEASE
```

CORE_SUSPEND_SWITCH_RELEASE

Description Suspends any switch releases to be passed to the application.

Usage CORE_SUSPEND_SWITCH_RELEASE

Assumptions Any of the following switches -- `bCOREModeSwitch, bCOREStopResetSwitch, bCOREStartSplitSwitch` -- must be allowed in the system. Refer to the

CORE_ALLOW_KEYS for activating these switches.

This is the default condition whenever a mode or state change occurs.

Input None

Output None

Destroys HL

Example
```
;do not pass the switch release events to the foreground application
CORE_SUSPEND_SWITCH_RELEASE
```

CORE_ENABLE_GENERIC_SWITCH_EVENT

Description Request the system to generate a generic switch depress or release event for application processing instead of sending specific switch event.

This will convert the following system events into the generic events COREEVENT_ANYSWITCHDEPRESS and COREEVENT_ANYSWITCHRELEASE.

```
COREEVENT_SWITCH1DEPRESS  }
COREEVENT_SWITCH2DEPRESS  } --> COREEVENT_ANYSWITCHDEPRESS
COREEVENT_SWITCH3DEPRESS  }

COREEVENT_SWITCH1RELEASE  }
COREEVENT_SWITCH2RELEASE  } --> COREEVENT_ANYSWITCHRELEASE
COREEVENT_SWITCH3RELEASE  }
```

Usage CORE_ENABLE_GENERIC_SWITCH_EVENT

Assumptions Any of the following switches -- bCOREModeSwitch, bCOREStopResetSwitch, bCOREStartSplitSwitch -- must be allowed in the system. Refer to the CORE_ALLOW_KEYS for activating these switches.

Input None

Output None

Destroys HL

Example
```
; process state entry
ld  A, [CORECurrentEvent]
cp  A, #COREEVENT_STATEENTRY
jr  NZ, check_next_event1

;request for a generic switch event. We do this since by default, the
;system will always pass specific switch events
CORE_ENABLE_GENERIC_SWITCH_EVENT

; we would like to handle only the mode and stop/reset switch
CORE_ALLOW_KEYS (bCOREModeSwitch|bCOREStopResetSwitch)
```

```
   ret

check_next_event1:

   cp  A, #COREEVENT_ANYSWITCHDEPRESS
   jr  NZ, check_next_event2

   ; TODO: write code to handle any switch depression
   ret

check_next_event2:

   …
```

CORE_DISABLE_GENERIC_SWITCH_EVENT

Description Request the system to generate a specific switch depress or release events for application processing.

Usage CORE_DISABLE_GENERIC_SWITCH_EVENT

Assumptions Any of the following switches -- `bCOREModeSwitch, bCOREStopResetSwitch, bCOREStartSplitSwitch` -- must be allowed in the system. Refer to the CORE_ALLOW_KEYS for activating these switches.

This is the default condition whenever a mode or state change occurs.

Input None

Output None

Destroys HL

Example

```
   ; process state entry
   ld  A, [CORECurrentEvent]
   cp  A, #COREEVENT_STATEENTRY
   jr  NZ, check_next_event1

   ;request for a generic switch event. We do not need to do this
   ;since by default, the system will always pass specific switch events,
   ;but is shown here for reference to differentiate between ENABLE and
   ;DISABLE GENERIC SWITCH EVENT APIs
   CORE_DISABLE_GENERIC_SWITCH_EVENT

   ; we would like to handle only the mode and stop/reset switch
   CORE_ALLOW_KEYS (bCOREModeSwitch|bCOREStopResetSwitch)
   ret

check_next_event1:

   cp  A, #COREEVENT_MODESWITCHDEPRESS
   jr  NZ, check_next_event2

   CORE_REQ_MODE_CHANGE_NEXT
   ret
```

```
check_next_event2:

   cp  A, #COREEVENT_STOPRESETSWITCHDEPRESS
   jr  NZ, check_next_event3

   CORE_REQ_MODE_CHANGE_NEXT
   ret

check_next_event3:
   …
```

CORE_REQUEST_MELODY_POPUP_CANCEL

Description Request the system to send the system event COREEVENT_MELODYPOPUPCANCEL when any active melody during a popup is to be cancelled by any switch depression.

This facilitates popup melody cancellation using a single system event rather than handling all the different switch depression events. The switch depression event that was used to cancel the popup melody can be accessed from the variable COREEventArgument.

This is a one time request. Once the COREEVENT_MELODYPOPUPCANCEL event is passed to the foreground application, the system will automatically cancel the request.

Usage CORE_REQUEST_MELODY_POPUP_CANCEL

Assumptions None

Input None

Output None

Destroys None

Example

```
;request that the popup melody be cancelled by any switch (that was
;previously allowed by the key mask) and pass the event MELODYPOPUPCANCEL to
;the foreground application with the actual switch in the COREEventArgument
;variable
CORE_REQUEST_MELODY_POPUP_CANCEL
…


   ld  A,  [CORECurrentEvent]
   cp  A,  #COREEVENT_MELODYPOPUPCANCEL
   jr  NZ, CheckNextEvent1

   ld  A,  [COREEventArgument]
   cp  A,  #COREEVENT_MODESWITCHDEPRESS
   jr  Z,  ModeSwitchWasUsedToCancelPopupMelody

CheckNextEvent1:
```

CORE_ENABLE_PULSE_MODE

Description Sets the system to send ring pulse events to the foreground application instead of ring edge events on crown rotation.

Usage CORE_ENABLE_PULSE_MODE

Assumptions Any of the following switches – `bCORECWSwitch, bCORECCWSwitch` must be allowed in the system. Refer to the CORE_ALLOW_KEYS for activating these switches.

Input None

Output None

Destroys HL, A, B

Example

```
;enable pulses to be sent to the foreground application rather than edges
CORE_ENABLE_PULSE_MODE

…

   ld  A,  [CORECurrentEvent]
   cp  A,  #COREEVENT_CW_PULSES
   jr  NZ, CheckNextEvent1

   ;process CW pulses
   ld  B,  [COREEventArgument]
   ;B = number of CW pulses detected

CheckNextEvent1:

   cp  A,  #COREEVENT_CCW_PULSES
   jr  NZ, CheckNextEvent2

   ;process CCW pulses
   ld  B,  [COREEventArgument]
   ;B = number of CCW pulses detected

CheckNextEvent2:
```

CORE_DISABLE_PULSE_MODE

Description Sets the system to send ring edge events to the foreground application instead of ring pulse events on crown rotation.

Usage CORE_DISABLE_PULSE_MODE

Assumptions Any of the following switches – `bCORECWSwitch, bCORECCWSwitch` must be allowed in the system. Refer to the CORE_ALLOW_KEYS for activating these switches.

This is the default condition whenever a mode or state change occurs.

Input None

Output None

Destroys HL, A, B

Example

```
;enable edge to be sent to the foreground application rather than edges
CORE_DISABLE_PULSE_MODE
CORE_SUSPEND_RING_LEADING_EVENTS

…

   ld  A,  [CORECurrentEvent]
   cp  A,  #COREEVENT_CW_EDGE_TRAILING
   jr  NZ, CheckNextEvent1

   ;process CW trailing edge

CheckNextEvent1:

   cp  A,  #COREEVENT_CCW_EDGE_TRAILING
   jr  NZ, CheckNextEvent2

   ;process CCW trailing edge

CheckNextEvent2:
```

HW_KBD_CANCEL_CURRENT_SWITCH_RELEASE

HW_KBD_CANCEL_CURRENT_SWITCH_RELEASE

Description Indicates to the keyboard hardware driver not to pass currently depressed switch's release event to the core.

Usage HW_KBD_CANCEL_CURRENT_SWITCH_RELEASE

Assumptions

Input None

Output None

Destroys HL

Example

```
   ld  A,  [CORECurrentEvent]
   cp  A,  #COREEVENT_STARTSPLITDEPRESS
   jr  NZ, CheckNextEvent1

   ;process START/SPLIT depress

   ;we do not want to handle the release of this switch depression
```

```
    ;at this time…
    HW_KBD_CANCEL_CURRENT_SWITCH_RELEASE

CheckNextEvent1:

    …
```

Background Task Operations

CORE_REQ_BACKGROUND_TASK

Description Request a background task to be initiated by the system.

Usage CORE_REQ_BACKGROUND_TASK

Assumptions None

Input

L - Application Index
A - System Event to be passed to the background handler routine for processing

The kernel provides the following constants for system events to be processed in the background handler:

Constant
COREEVENT_PRIMARY_TIME_CHANGE
COREEVENT_REFRESH_START
COREEVENT_REFRESH_CONTINUE
COREEVENT_PEEK_SEARCH_START
COREEVENT_PEEK_SEARCH_CONTINUE
COREEVENT_DAY_UPDATE_START
COREEVENT_DAY_UPDATE_CONTINUE
COREEVENT_HOUR_UPDATE_START
COREEVENT_HOUR_UPDATE_CONTINUE
COREEVENT_NULL

Output None

Destroys None

Example

```
    ;background handler processing
    ld  A,  [COREBackgroundEvent]
    cp  A,  #COREEVENT_REFRESH_START
    jr  NZ, checknextevent1

    ;todo: write code here to process refresh event initialization

    ; we have determined that the handler needs more time to process
    ; request for a callback whenever the system is not busy.  Use the
    ; CONTINUE event…
    ld  L, [COREBackgroundAppIndex]
    ld  A, #COREEVENT_REFRESH_CONTINUE
```

```
   CORE_REQ_BACKGROUND_TASK
   ret

checknextevent1:
   …
```

CORE_REQ_BACKGROUND_TASK_WITH_PRIORITYCHECK

Description Request a background task to be initiated. A pending background task will not be overwritten if it has a higher priority setting compared to the new requested task.

Usage CORE_REQ_BACKGROUND_TASK_WITH_PRIORITYCHECK

Assumptions None

Input

L - Application Index
A - System Event to be passed to the background handler routine for processing

The kernel provides the following constants for system events to be processed in the background handler sorted according to descending priority:

Constant
COREEVENT_PRIMARY_TIME_CHANGE
COREEVENT_REFRESH_START
COREEVENT_REFRESH_CONTINUE
COREEVENT_PEEK_SEARCH_START
COREEVENT_PEEK_SEARCH_CONTINUE
COREEVENT_DAY_UPDATE_START
COREEVENT_DAY_UPDATE_CONTINUE
COREEVENT_HOUR_UPDATE_START
COREEVENT_HOUR_UPDATE_CONTINUE
COREEVENT_NULL

Output None

Destroys None

Example

```
DayUpdateProcessing:

   ld  B, [UpdateFlag]
   bit B, #bDayUpdate
   jr  NZ, HourUpdateProcessing

   ld  L, [COREBackgroundAppIndex]
   ld  A, #COREEVENT_DAYUPDATE_START
   CORE_REQ_BACKGROUND_TASK_WITH_PRIORITYCHECK

HourUpdateProcessing:

   bit B, #bHourUpdate
   jr  NZ, MinuteUpdateProcessing

   ; do not request a new background task if the DayUpdate code has been
```

```
   ; executed
   ld  L, [COREBackgroundAppIndex]
   ld  A, #COREEVENT_HOURUPDATE_START
   CORE_REQ_BACKGROUND_TASK_WITH_PRIORITYCHECK

MinuteUpdateProcessing:

   bit B, #bMinuteUpdate
   jr  NZ, UpdateProcessingEnd

   ; do not request a new background task if the DayUpdate and HourUpdate
   ; code has been executed
   ld  L, [COREBackgroundAppIndex]
   ld  A, #COREEVENT_PEEKSEARCH_START
   CORE_REQ_BACKGROUND_TASK_WITH_PRIORITYCHECK

UpdateProcessingEnd:
```

CORE_REQ_BACKGROUND_TASK_FOR_APPTYPE

Description Request a background task to be initiated for all active application that matches the specified application type.

Usage CORE_REQ_BACKGROUND_TASK_FOR_APPTYPE

Assumptions None

Input

A - Application Type
L - System Event to be passed to the background handler routine for processing

The kernel provides the following constants for system events to be processed in the background handler:

Constant
COREEVENT_PRIMARY_TIME_CHANGE
COREEVENT_REFRESH_START
COREEVENT_REFRESH_CONTINUE
COREEVENT_PEEK_SEARCH_START
COREEVENT_PEEK_SEARCH_CONTINUE
COREEVENT_DAY_UPDATE_START
COREEVENT_DAY_UPDATE_CONTINUE
COREEVENT_HOUR_UPDATE_START
COREEVENT_HOUR_UPDATE_CONTINUE
COREEVENT_NULL

The kernel provides some built-in constants for application type:

Constant
COREAPPTYPESYSTEM
COREAPPTYPECOMM
COREAPPTYPEOPTION
COREAPPTYPETOD

```
COREAPPTYPEDATE
COREAPPTYPECHRONO
COREAPPTYPETIMER
COREAPPTYPESYNCHROTIMER
COREAPPTYPECOUNTER
COREAPPTYPECONTACT
COREAPPTYPETASK
COREAPPTYPENOTES
COREAPPTYPESCHEDULE
COREAPPTYPETIDE
COREAPPTYPEDEMO
COREAPPTYPEGAME
COREAPPTYPEALARM
COREAPPTYPEAPPOINTMENT
COREAPPTYPEOCCASION
```

Output None

Destroys None

Example

```
todSettingComplete:

   ;since we have updated the primary time zone, we
   ;have to alert alarm and appointment active in the system
   ;with this change

   ; tell any alarm application in system about the change
   ld  A, #COREAPPTYPEALARM
   ld  L, #COREEVENT_PRIMARY_TIME_CHANGE
   CORE_REQ_BACKGROUND_TASK_FOR_APPTYPE

   ; tell any appointment application in system about the change
   ld  A, #COREAPPTYPEAPPOINTMENT
   ld  L, #COREEVENT_PRIMARY_TIME_CHANGE
   CORE_REQ_BACKGROUND_TASK_FOR_APPTYPE
```

CORE_REQ_BACKGROUND_TASK_FOR_APPTYPE_WITH_PRIORITYCHECK

Description Request a background task to be initiated for the specified application type but checks first before overwriting a higher priority task.

Usage CORE_REQ_BACKGROUND_TASK_FOR_APPTYPE_WITH_PRIORITYCHECK

Assumptions None

Input

A - Application Type
L - System Event to be passed to the background handler routine for processing

The kernel provides the following constants for system events to be processed in the background handler sorted according to descending priority:

Constant

```
COREEVENT_PRIMARY_TIME_CHANGE
COREEVENT_REFRESH_START
COREEVENT_REFRESH_CONTINUE
COREEVENT_PEEK_SEARCH_START
COREEVENT_PEEK_SEARCH_CONTINUE
COREEVENT_DAY_UPDATE_START
COREEVENT_DAY_UPDATE_CONTINUE
COREEVENT_HOUR_UPDATE_START
COREEVENT_HOUR_UPDATE_CONTINUE
COREEVENT_NULL
```

The kernel provides some built-in constants for application type:

Constant
COREAPPTYPESYSTEM
COREAPPTYPECOMM
COREAPPTYPEOPTION
COREAPPTYPETOD
COREAPPTYPEDATE
COREAPPTYPECHRONO
COREAPPTYPETIMER
COREAPPTYPESYNCHROTIMER
COREAPPTYPECOUNTER
COREAPPTYPECONTACT
COREAPPTYPETASK
COREAPPTYPENOTES
COREAPPTYPESCHEDULE
COREAPPTYPETIDE
COREAPPTYPEDEMO
COREAPPTYPEGAME
COREAPPTYPEALARM
COREAPPTYPEAPPOINTMENT
COREAPPTYPEOCCASION

Output None

Destroys None

Example

```
;request a background task for any alarm type application
ld  A, #COREAPPTYPEALARM
ld  L, # COREEVENT_REFRESH_START
CORE_REQ_BACKGROUND_TASK_FOR_APPTYPE_WITH_PRIORITYCHECK
```

CORE_REQ_BACKGROUND_TASK_FOR_PTZBASEDAPPS

Description Request a background task to be initiated for application types that are dependent upon the primary time zone data.

These application types are usually dependent on the primary time zone:

Constant

```
COREAPPTYPEALARM
COREAPPTYPEAPPOINTMENT
COREAPPTYPEOCCASION
```

By definition, application type with indexes from 0xE0 to 0xFF are considered dependent on the primary time zone.

Usage CORE_REQ_BACKGROUND_TASK_FOR_PTZBASEDAPPS

Assumptions None

Input L - System Event to be passed to the background handler routine for processing

The kernel provides the following constants for system events to be processed in the background handler sorted according to descending priority:

Constant
COREEVENT_PRIMARY_TIME_CHANGE
COREEVENT_REFRESH_START
COREEVENT_REFRESH_CONTINUE
COREEVENT_PEEK_SEARCH_START
COREEVENT_PEEK_SEARCH_CONTINUE
COREEVENT_DAY_UPDATE_START
COREEVENT_DAY_UPDATE_CONTINUE
COREEVENT_HOUR_UPDATE_START
COREEVENT_HOUR_UPDATE_CONTINUE
COREEVENT_NULL

Output None

Destroys None

Example

```
todSettingComplete:

   ;since we have updated the primary time zone, we
   ;have to alert all applications dependent on the primary time zone
   ld  L, #COREEVENT_PRIMARY_TIME_CHANGE
   CORE_REQ_BACKGROUND_TASK_FOR_PTZBASEDAPPS
```

CORE_CLEAR_BACKGROUND_APPLICATION_TASK

Description Clears the pending background application task for the specified application index.

Usage CORE_CLEAR_BACKGROUND_APPLICATION_TASK

Assumptions None

Input L - Application Index

Output None

Destroys None

Example
```
ld  L, [CORECurrentMode]
CORE_CLEAR_BACKGROUND_APPLICATION_TASK
```

CORE_CLEAR_ALL_BACKGROUND_APPLICATION_TASK

Description Clears all pending background application task.

Usage CORE_CLEAR_ALL_BACKGROUND_APPLICATION_TASK

Assumptions None

Input None

Output None

Destroys None

Example
```
;clear all background task
CORE_CLEAR_ALL_BACKGROUND_APPLICATION_TASK
```

Popup Operations

CORE_REQ_POPDOWN

Description Request a popdown operation. Once processed, the system will restore the interrupted foreground application. The state return is usually the interrupt state index unless the CORE_SET_POPDOWN_STATE specified a different return state.

Usage CORE_REQ_POPDOWN

Assumptions This must be called only in a popup state handler. Using this API on a non-popup state will produce unpredictable results or even a reset.

Input None

Output None

Destroys B

Example
```
;completed processing the popup, request for a popdown
CORE_REQ_POPDOWN
ret
```

CORE_SET_POPDOWN_STATE

Description Sets the return state index when the system returns from a popup.

This is usually called in the state entry event handler to ate that will indicate to the core to popdown to a different state if a popup occurs

Usage CORE_SET_POPDOWN_STATE {*popdown_state*}

Assumptions None

Input *popdown_state* - State index of the current application to proceed on a popdown.

Output None

Destroys B

Example
```
;application banner state handler
;if a popup occurs in this state, it should proceed to the default
;state on a popup
CORE_SET_POPDOWN_STATE COREDEFAULTSTATE
```

CORE_ENABLE_POPUPS

Description Allow popups to occur in the current state. On exit from the popup, it will return to the interrupted state unless CORE_SET_POPDOWN_STATE was executed prior to the popup.

Usage CORE_ENABLE_POPUPS

Assumptions None

Input None

Output None

Destroys None

Example
```
;allow popups to occur in the current state handler
CORE_ENABLE_POPUPS
```

CORE_SUSPEND_POPUPS

Description Suspend any popups from occurring in the current state.

Usage CORE_SUSPEND_POPUPS

Assumptions None

Input None

Output None

Destroys None

Example
```
;do not allow any popups to occur in the current state handler
CORE_SUSPEND_POPUPS
```

CORE_CANCEL_ALL_POPUPS

Description Cancel all pending popups.

Usage CORE_CANCEL_ALL_POPUPS

Assumptions None

Input None

Output None

Destroys A, HL

Example
```
;cancel all popups currently being queued in the system
CORE_CANCEL_ALL_POPUPS
```

CORE_CANCEL_ANY_PTZ_POPUPS

Description Cancels all popups queued that is dependent on the primary time zone. These popups are queued under the Time Zone Check resource and the Backup resource.

Usage CORE_CANCEL_ANY_PTZ_POPUPS

Assumptions None

Input None

Output None

Destroys A, B, HL

Example
```
;cancel pending popups of application dependent on the primary time zone
CORE_CANCEL_ANY_PTZ_POPUPS
```

CORETASK_POPUP_REQUEST_BCK

Description Request for a popup to be queued under the Backup Resource.

Usage CORETASK_POPUP_REQUEST_BCK

Assumptions None

Input
A - Application Index
L - Backup Resource Index

Output None

Destroys A, HL, IY

Example
```
;** called in the background handler**
;request a popup to the owner of the backup resource
ld  A, [COREBackgroundAppIndex]
ld  L, #0   ;resource 0
CORE_TASK_POPUP_REQUEST_BCK
```

CORETASK_POPUP_CANCEL_BCK

Description Cancel popup request queued under a Backup Resource

Usage CORETASK_POPUP_CANCEL_BCK

Assumptions None

Input
L - Backup Resource Index

Output None

Destroys A, HL, IY

Example
```
;** called in the background handler**
;cancel pending backup resource popup
ld  L, #0   ;resource 0
CORE_TASK_POPUP_CANCEL_BCK
```

CORETASK_POPUP_REQUEST_TZC

Description Request for a popup to be queued under the Time Zone Check Resource.

Usage CORETASK_POPUP_REQUEST_TZC

Assumptions None

Input
A - Application Index
L - Time Zone Check Resource Index

Output None

Destroys A, HL, IY

Example

```
;** called in the background handler**
;request a popup to the owner of the time zone check resource
ld  A, [COREBackgroundAppIndex]
ld  L, #0   ;resource 0
CORE_TASK_POPUP_REQUEST_TZC
```

CORETASK_POPUP_CANCEL_TZC

Description Cancel popup request queued under a Time Zone Check Resource

Usage CORETASK_POPUP_CANCEL_TZC

Assumptions None

Input
L - Time Zone Check Resource Index

Output None

Destroys A, HL, IY

Example

```
;** called in the background handler**
;cancel pending time zone resource popup
ld  L, #0   ;resource 0
CORE_TASK_POPUP_CANCEL_TZC
```

CORETASK_POPUP_REQUEST_TMR

Description Request for a popup to be queued under the Timer Resource.

Usage CORETASK_POPUP_REQUEST_TMR

Assumptions None

Input
A - Application Index
L - Timer Resource Index

Output None

Destroys A, HL, IY

Example
```
;** called in the background handler**
;request a popup to the owner of the timer resource
ld  A, [COREBackgroundAppIndex]
ld  L, #0   ;resource 0
CORE_TASK_POPUP_REQUEST_TMR
```

CORETASK_POPUP_CANCEL_TMR

Description Cancel popup request queued under a Timer Resource

Usage CORETASK_POPUP_CANCEL_TMR

Assumptions None

Input
L - Timer Resource Index

Output None

Destroys A, HL, IY

Example
```
;** called in the background handler**
;cancel pending timer resource popup
ld  L, #0   ;resource 0
CORE_TASK_POPUP_CANCEL_TMR
```

CORETASK_POPUP_REQUEST_STP

Description Request for a popup to be queued under the Stopwatch Resource.

Usage CORETASK_POPUP_REQUEST_STP

Assumptions None

Input
A - Application Index
L - Stopwatch Resource Index

Output None

Destroys A, HL, IY

Example
```
;** called in the background handler**
;request a popup to the owner of the stopwatch resource
ld  A, [COREBackgroundAppIndex]
ld  L, #0   ;resource 0
CORE_TASK_POPUP_REQUEST_STP
```

CORETASK_POPUP_CANCEL_STP

Description Cancel popup request queued under a Stopwatch Resource

Usage CORETASK_POPUP_CANCEL_STP

Assumptions None

Input L - Stopwatch Resource Index

Output None

Destroys A, HL, IY

Example
```
;** called in the background handler**
;cancel pending stopwatch resource popup
ld  L, #0   ;resource 0
CORE_TASK_POPUP_CANCEL_STP
```

CORETASK_POPUP_REQUEST_SYN

Description Request for a popup to be queued under the Synchro Resource.

Usage CORETASK_POPUP_REQUEST_SYN

Assumptions None

Input A - Application Index
L - Synchro Resource Index

Output None

Destroys A, HL, IY

Example
```
;** called in the background handler**
;request a popup to the owner of the synchro resource
ld  A, [COREBackgroundAppIndex]
```

```
ld  L, #0   ;resource 0
CORE_TASK_POPUP_REQUEST_SYN
```

CORETASK_POPUP_CANCEL_SYN

Description Cancel popup request queued under a Synchro Resource

Usage CORETASK_POPUP_CANCEL_SYN

Assumptions None

Input L - Synchro Resource Index

Output None

Destroys A, HL, IY

Example

```
;** called in the background handler**
;cancel pending synchro resource popup
ld  L, #0   ;resource 0
CORE_TASK_POPUP_CANCEL_SYN
```

Utilities

CORE_ENABLE_SET_MODE

Description Sets a general purpose flag usually used to indicate that the application is currently in a set mode.

Usage CORE_ENABLE_SET_MODE

Assumptions None

Input None

Output None

Destroys HL

Example

```
;set flag
CORE_ENABLE_SET_MODE
```

CORE_DISABLE_SET_MODE

Description Clears a general purpose flag usually used to indicate that the application is not in a set mode.

Usage CORE_DISABLE_SET_MODE

Assumptions None

Input None

Output None

Destroys HL

Example
```
;clear flag
CORE_DISABLE_SET_MODE
```

CORE_SET_RANDOM_SEED

Description Sets the base number for the pseudo-random number generator.

Usage CORE_SET_RANDOM_SEED

Assumptions None

Input None

Output None

Destroys None

Example
```
;get a new seed for the random number generator
CORE_SET_RANDOM_SEED
```

CORE_GENERATE_RANDOM_NUMBER

Description Generates a pseudo-random number from 0 – 255.

Usage CORE_GENERATE_RANDOM_NUMBER

Assumptions None

Input None

Output A - Psuedo-Random number from 0-255. The new number is based on the old random number generated.

Destroys None

Example
```
;get a random number with a new seed
CORE_SET_RANDOM_SEED
CORE_GENERATE_RANDOM_NUMBER

cp   A,  #128
jr   C,  RandomNumberIsLessThan128
jr   NC, RandomNumberIsGreaterThanOrEqual128
```

CORE_DATABASE_MODIFIED_BY_USER

Description Updates the flag that the application database was modified by the user. This flag is checked by the PC to determine if it should include it in its list to upload the database.

Usage CORE_DATABASE_MODIFIED_BY_USER

Assumptions None

Input A - Application Index

Output None

Destroys None

Example
```
;database has been modified.  Set the falg to alert the PC during upload
ld  A, [CORECurrentMode]
CORE_DATABASE_MODIFIED_BY_USER
```

CORE_APP_ID_PRESENT

Description Checks if a given an application ID (Type and Instance) is presently active in the system.

Usage CORE_APP_ID_PRESENT

Assumptions The application during design time must know beforehand what the instance number of an application. Though there are other ways (documented and undocumented) to get the instance number to be used for this function. This might end up to be a least used utility.

Input IX - Application ID
LOWBYTE(IX) - Application Type
HIBYTE(IX) - Application Instance

The kernel provides some built-in constants for application type:

Constant

COREAPPTYPESYSTEM
COREAPPTYPECOMM
COREAPPTYPEOPTION
COREAPPTYPETOD
COREAPPTYPEDATE
COREAPPTYPECHRONO
COREAPPTYPETIMER
COREAPPTYPESYNCHROTIMER
COREAPPTYPECOUNTER
COREAPPTYPECONTACT
COREAPPTYPETASK
COREAPPTYPENOTES
COREAPPTYPESCHEDULE
COREAPPTYPETIDE
COREAPPTYPEDEMO
COREAPPTYPEGAME
COREAPPTYPEALARM
COREAPPTYPEAPPOINTMENT
COREAPPTYPEOCCASION

Output None
Success: A = Application Index
Failure: A = 0xFF

Destroys None

Example
```
;check if a specific ID is present in the systm
ld  IX, #( (COREAPPTYPECHRONO << 8) | 00H )
CORE_APP_ID_PRESENT
cp  A, #0FFH
jr  Z,  RequestedAppIDIsNotPresent
```

CORE_GET_APP_INDEX

Description Checks if a given an application ID (Type and Instance) is presently active in the system. The same functionality as CORE_APP_ID_PRESENT.

Usage CORE_GET_APP_INDEX

Assumptions The application during design time must know beforehand what the instance number of an application. Though there are other ways (documented and undocumented) to get the instance number to be used for this function. This might end up to be a least used utility.

Input IX - Application ID
LOWBYTE(IX) - Application Type
HIBYTE(IX) - Application Instance

The kernel provides some built-in constants for application type:

Constant
COREAPPTYPESYSTEM

```
COREAPPTYPECOMM
COREAPPTYPEOPTION
COREAPPTYPETOD
COREAPPTYPEDATE
COREAPPTYPECHRONO
COREAPPTYPETIMER
COREAPPTYPESYNCHROTIMER
COREAPPTYPECOUNTER
COREAPPTYPECONTACT
COREAPPTYPETASK
COREAPPTYPENOTES
COREAPPTYPESCHEDULE
COREAPPTYPETIDE
COREAPPTYPEDEMO
COREAPPTYPEGAME
COREAPPTYPEALARM
COREAPPTYPEAPPOINTMENT
COREAPPTYPEOCCASION
```

Output None
Success: A = Application Index
Failure: A = 0xFF

Destroys None

Example
```
;check if a specific ID is present in the systm
ld  IX, #( (COREAPPTYPECHRONO << 8) | 00H )
CORE_GET_APP_INDEX
cp  A, #0FFH
jr  Z,  RequestedAppIDIsNotPresent
```

CORE_GET_APP_INDEX_FROM_APP_TYPE

Description Checks if a given Application Type is currently active in the system. If more than one application matches the same type, the first application found based on the mode list table will be selected.

Usage CORE_GET_APP_INDEX_FROM_APP_TYPE

Assumptions None

Input A - Application Type

The kernel provides some built-in constants for application type:

Constant
COREAPPTYPESYSTEM
COREAPPTYPECOMM
COREAPPTYPEOPTION
COREAPPTYPETOD
COREAPPTYPEDATE
COREAPPTYPECHRONO

```
COREAPPTYPETIMER
COREAPPTYPESYNCHROTIMER
COREAPPTYPECOUNTER
COREAPPTYPECONTACT
COREAPPTYPETASK
COREAPPTYPENOTES
COREAPPTYPESCHEDULE
COREAPPTYPETIDE
COREAPPTYPEDEMO
COREAPPTYPEGAME
COREAPPTYPEALARM
COREAPPTYPEAPPOINTMENT
COREAPPTYPEOCCASION
```

Output None
Success: A = Application Index
Failure: A = 0xFF

Destroys None

Example

```
;get the first app index of an appointment type application
ld   A, #COREAPPTYPEAPPOINTMENT
CORE_GET_APP_INDEX_FROM_APP_TYPE
cp   A,#0FFH
jr   Z, ThereIsNoAppointmentApplicationPresentInSystem
```

AUDIO API

These are used to handle melody generation.

AUDSTART_SYSTEM_MELODY

Description Starts generation of a system melody.

Usage AUDSTART_SYSTEM_MELODY {*melody_index*}, {*melody_done_action*}

Assumptions None

Input

melody_index	-	system melody to generate
melody_done_action	-	Action to be done when melody generation is complete.

The following constants are available for *melody_index*:

Constant	Description
AUDSWBEEPMELODY	Switch beep melody
AUDHOURCHIMEMELODY	Hour Chime melody
AUDALARMMELODY	Alarm melody
AUDAPPOINTMENTMELODY	Appointment melody
AUDTIMERMELODY	Timer Melody
AUDINTERVALTIMERMELODY	Interval Timer Melody
AUDHALFTIMERMELODY	Half timer melody
AUDCOMMERRORMELODY	Communication Error melody
AUDCUSTOMMELODY	Custom melody

The following constants are available for *melody_done_action*:

Constant	Description
AUDSENDMELODYDONEEVENT	Send COREEVENT_MELODY_DONE at the end of melody to application.
AUDNOMELODYDONEEVENT	Do not send COREEVENT_MELODY_DONE at the end of melody to application.

Output None

Destroys BA

Example

```
;request for the alarm melody to be generated and send an event when the
;entire melody pattern has been completed
AUDSTART_SYSTEM_MELODY AUDALARMMELODY, AUDSENDMELODYDONEEVENT
```

AUDSETUP_MELODYADDRESS

Description Setup the base address of a melody into the melody table. Usually used to setup a new melody pattern for the custom melody offset.

Usage AUDSETUP_MELODYADDRESS

Assumptions None

Input
HL - base memory address of where the melody pattern is located
BA - melody table index to setup

The following constants are available for BA:

Constant	Description
AUDSWBEEPMELODY	Switch beep melody
AUDHOURCHIMEMELODY	Hour Chime melody
AUDALARMMELODY	Alarm melody
AUDAPPOINTMENTMELODY	Appointment melody
AUDTIMERMELODY	Timer Melody
AUDINTERVALTIMERMELODY	Interval Timer Melody
AUDHALFTIMERMELODY	Half timer melody
AUDCOMMERRORMELODY	Communication Error melody
AUDCUSTOMMELODY	Custom melody

Output None

Destroys IX

Example

```
;Generate a custom melody

;setup the address of the user melody
ld  HL, #MyNewMelodyPatternAddress
ld  BA, #AUDCUSTOMMELODY
AUDSETUP_MELODYADDRESS

;start generating the user melody
AUDSTART_SYSTEM_MELODY AUDCUSTOMMELODY, AUDNOMELODYDONEEVENT
```

AUDSTOP_MELODY

Description Stops any melody currently being generated.

Usage AUDSTOP_MELODY

Assumptions None

Input None

Output None

Destroys HL

Example

```
;stop any current melody being generated
AUDSTOP_MELODY
```

DATABASE API

These are used to handle accesses to data or code stored in external memory.

Open and Close Database Operations

DB_OPEN_FILE

Description Open a database file in external memory for read and write access.
The EEPROM is powered up before the file is opened.

Usage DB_OPEN_FILE

Assumptions The database file opened must follow the database header structure requirements. This macro should not be used to open a link-list type of database. Use DB_OPEN_FILE_LINK_LIST instead.

Input

DBExternalMemoryAddress	-	16-bit absolute address of the database file in external memory

Output

DBExternalBaseAddress	-	Base address of database use to convert offsets to absolute address (16-bit). Used only by the database drivers.
DBNumberOfRecords	-	Number of records stored for the current database (16-bit).
DBSizePerRecord	-	Size of record of database. Significant only for fixed-size random access database.

Destroys None

Example

```
; Open the associated database of the current application
ld      HL, [CORECurrentADDAddress]
ld      [DBExternalMemoryAddress], HL
DB_OPEN_FILE

; get the number of records of the opened database
ld      HL, [DBNumberOfRecords]
ld      [CONTTotalNumberOfRecords], HL
DB_CLOSE_FILE
```

DB_OPEN_FILE_LINK_LIST

Description Open a linked-list database file in external memory for read and write access.
The EEPROM is powered up before the file is opened.

Usage DB_OPEN_FILE_LINK_LIST

Assumptions The database file opened must follow the database header structure requirements. This macro should not be used to open a non-link-list type of database. Use DB_OPEN_FILE instead.

Input

DBExternalMemoryAddress	-	16-bit absolute address of the database file in external memory

Output

DBExternalBaseAddress	-	Base address of database use to convert offsets to absolute

address (16-bit). Used only by the database drivers.
DBNumberOfRecords - Number of records stored for the current database (16-bit).

Destroys None

Example
```
; Open the associated linked list database of the current application
ld      HL, [CORECurrentADDAddress]
ld      [DBExternalMemoryAddress], HL
DB_OPEN_FILE_LINK_LIST
```

DB_CLOSE_FILE

Description Close the currently opened database file. The EEPROM is powered down.

Usage DB_CLOSE_FILE

Assumptions None

Input None

Output None

Destroys None

Example
```
;close any database currently opened.  This will remove power
;from the eeprom
DB_CLOSE_FILE
```

Database Information Header Operations

DB_READ_APPLICATION_INFOHEADER

Description Reads the application specific data from a database header.
This macro will powerup and powerdown the EEPROM.

This is usually used during **`COREEVENT_INIT`** processing in the background handler. When the PC downloads a database to the watch, it passes information about the database in the header section of the database. For example, the PC will specify in this section the number of records stored in the database. This will be used by the WristApp for its internal processing to prevent it from writing records beyond the specified number of records.

Usage DB_READ_APPLICATION_INFOHEADER

Assumptions The database structure must conform to the M851 Database Structure requirements.

Input DBExternalMemoryAddress - absolute address of database in external memory (16-bit)

DBInternalMemoryAddress	-	destination address in internal memory (16-bit)
DBLengthLo	-	number of bytes to read (8-bit)
DBRecordOffset	-	from start of app specific header to begin reading (8-bit)

Output None

Destroys None

Example

```
; Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE

; setup the absolute address of the current application's database
ld      IY, [COREBackgroundADDAddress]
ld      [DBExternalMemoryAddress],  IY

; setup the internal memory to store the database header information
ld  HL, #MyApplicationHeaderBaseAddress
ld  [DBInternalMemoryAddress], HL

; setup the number of bytes to read from the header information
ld  A, #MyApplicationHeaderSize
ld  [DBLengthLo], A

; setup the offset from the start of the header information to begin reading
ld  A, #0
ld  [DBRecordOffset], A

; read the header information
DB_READ_APPLICATION_INFOHEADER
```

DB_WRITE_APPLICATION_INFOHEADER

Description Updates the data in the application specific header of the database.
This macro will powerup and powerdown the EEPROM.

This is usually use during **COREEVENT_UPDATEDATABASEHEADER** in the background handler for WristApps that modifies database contents in the watch. This will update the header section so that a WristApp configuration application on the PC can decipher the changes made to the database. For example, changes made to a linked list database head pointer (deletion of the first record or insertion to the first record) needs to be communicated back to the PC configuration application so it can follow the new link list.

Usage DB_WRITE_APPLICATION_INFOHEADER

Assumptions The database structure must conform to the M851 Database Structure requirements.

Input

DBExternalMemoryAddress	-	absolute address of database in external memory (16-bit)
DBInternalMemoryAddress	-	source address in internal memory (16-bit)
DBLengthLo	-	number of bytes to write (8-bit)
DBRecordOffset	-	from start of app specific header to begin writing (8-bit)

Output None

Destroys None

Example

```
; Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE

; setup the absolute address of the current application's database
ld      IY, [COREBackgroundADDAddress]
ld      [DBExternalMemoryAddress],  IY

; setup the internal memory to copy into the database header information
ld  HL, #MyApplicationHeaderBaseAddress
ld  [DBInternalMemoryAddress], HL

; setup the number of bytes to write to the header information
ld  A, #MyApplicationHeaderSize
ld  [DBLengthLo], A

; setup the offset from the start of the header information to begin writing
ld  A, #0
ld  [DBRecordOffset], A

; write the updated header information back into the database
DB_WRITE_APPLICATION_INFOHEADER
```

DB_GET_ABSOLUTE_ADDRESS_OF_RECORD_RANDOMVAR

Description Returns the absolute address of a record in a variable-sized random access database.

Usage DB_GET_ABSOLUTE_ADDRESS_OF_RECORD_RANDOMVAR

Assumptions The database file has been previously opened.
The database must be a variable-sized random access database structure.
Record size is limited to 256 bytes.

Input

DBRecordNumber	-	record number to compute (16-bit)

Output

BA	-	absolute address of specified record in EEPROM (16-bit)
DBExternalMemoryAddress	-	absolute address of specified record in EEPROM (16-bit)

Destroys None

Example

```
; Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE

;get the absolute eeprom address of the specified record
ld  HL, [MyCurrentRecord]
ld  [DBRecordNumber], HL
DB_GET_ABSOLUTE_ADDRESS_OF_RECORD_RANDOMVAR
```

DB_GET_ABSOLUTE_ADDRESS_OF_RECORD_RANDOMFIX

Description Returns the absolute address of a record in a fixed-sized random access database.

Usage DB_GET_ABSOLUTE_ADDRESS_OF_RECORD_RANDOMFIX

Assumptions The database file has been previously opened.
The database must be a fixed-sized random access database structure.
Record size is limited to 256 bytes.

Input

DBRecordNumber	-	record number to compute (16-bit)

Output

BA	-	absolute address of specified record in EEPROM (16-bit)
DBExternalMemoryAddress	-	absolute address of specified record in EEPROM (16-bit)

Destroys None

Example

```
; Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE

;get the absolute eeprom address of the specified record
ld  HL, [MyCurrentRecord]
ld  [DBRecordNumber], HL
DB_GET_ABSOLUTE_ADDRESS_OF_RECORD_RANDOMFIX
```

Record Read and Write Operations

DB_WRITE_RECORD

Description Writes a specified record data to database in external memory starting at the start address of record. This is used for writing data to a sequential access and linked list database structure.

Usage DB_WRITE_RECORD

Assumptions The database file has been previously opened.
Record size is limited to 256 bytes.

Input

DBInternalMemoryAddress	-	source base address in internal memory (16-bit)
DBExternalMemoryAddress	-	destination record offset in external memory (16-bit)
DBLengthLo	-	number of bytes to write (8-bit)

Output None

Destroys None

Example

```
; Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE_LINK_LIST

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; write data from internal memory to eeprom
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; specify the base offset of a record in the database
ld  HL, [MyLinkListHeadPtr]
ld  [DBExternalMemoryAddress], HL

; specify the source data array to write to the database
ld  HL, #MyRecordDataBaseAddress
ld  [DBInternalMemoryAddress], HL

; specify the number of bytes to transfer
ld  A, #MyRecordDataSize
ld  [DBLengthLo], A

; write the data array to eeprom database
DB_WRITE_RECORD
```

DB_WRITE_RECORD_WITHOFFSET

Description Writes a specified record data to database in external memory starting at an offset from start address of record.

Usage DB_WRITE_RECORD_WITHOFFSET

Assumptions The database file has been previously opened.
Record size is limited to 256 bytes.

Input

DBInternalMemoryAddress	-	source base address in internal memory (16-bit)
DBExternalMemoryAddress	-	destination record offset in external memory (16-bit)
DBLengthLo	-	number of bytes to write (8-bit)
DBRecordOffset	-	offset from start address of record to begin writing (8-bit)

Output None

Destroys None

Example

```
; Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE_LINK_LIST

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; write data from internal memory to eeprom
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
; specify the base address of a record in the database
ld  HL, [MyLinkListHeadPtr]
ld  [DBExternalMemoryAddress], HL

; specify the source data array to write to the database
ld  HL, #MyRecordDataBaseAddress
ld  [DBInternalMemoryAddress], HL

; specify the number of bytes to transfer
ld  A, #MyRecordDataSize
ld  [DBLengthLo], A

; specify the byte offset from base address of a record
ld  A, #ActualStartOfRecordData
ld  [DBRecordOffset], A

; write the data array to eeprom database
DB_WRITE_RECORD_WITHOFFSET
```

DB_WRITE_RECORD_RANDOMFIX

Description Writes to a specific record in a fixed-sized random access database in external memory starting at the start address of record.

Usage DB_WRITE_RECORD_RANDOMFIX

Assumptions The database file has been previously opened.
The database structure must adhere to the header requirements of a random fixed database structure.
Record size is limited to 256 bytes.

Input

DBInternalMemoryAddress	-	source base address in internal memory (16-bit)
DBRecordNumber	-	Record number (16-bit)
DBLengthLo	-	number of bytes to write (8-bit)

Output None

Destroys None

Example

```
; Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE

; specify the parameters to write a data array to a random fix record
ld  HL, #MyRecordDataBaseAddress
ld  [DBInternalMemoryAddress], HL
ld  HL, [MyCurrentRecordNumber]
ld  [DBRecordNumber], HL
ld  A, #MyRecordDataSize
ld  [DBLengthLo], A

; write the data to eeprom
DB_WRITE_RECORD_RANDOMFIX
```

DB_WRITE_RECORD_WITHOFFSET_RANDOMFIX

Description Writes to a specific record in a fixed-sized random access database in external memory starting at a specified offset from the start address of record.

Usage DB_WRITE_RECORD_WITHOFFSET_RANDOMFIX

Assumptions The database file has been previously opened.
The database structure must adhere to the header requirements of a random fixed database structure.
Record size is limited to 256 bytes.

Input

DBInternalMemoryAddress	-	source base address in internal memory (16-bit)
DBRecordNumber	-	Record number (16-bit)
DBLengthLo	-	number of bytes to write (8-bit)
DBRecordOffset	-	offset from start address of record to begin writing (8-bit)

Output None

Destroys None

Example

```
; Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE

; specify the parameters to write a data array to a random fix record
ld  HL, #MyRecordDataBaseAddress
ld  [DBInternalMemoryAddress], HL
ld  HL, [MyCurrentRecordNumber]
ld  [DBRecordNumber], HL
ld  A, #MyRecordDataSize
ld  [DBLengthLo], A

; specify the offset into the record
ld  A, #MYRECORDDATAFIELD5OFFSET
ld  [DBRecordOffset], A

; write the data to eeprom
DB_WRITE_RECORD_WITHOFFSET_RANDOMFIX
```

DB_WRITE_RECORD_RANDOMVAR

Description Writes to a specific record in a variable-sized random access database in external memory starting at the start address of record.

Usage DB_WRITE_RECORD_RANDOMVAR

Assumptions The database file has been previously opened.

The database structure must adhere to the header requirements of a random variable size database structure.
Record size is limited to 256 bytes.

Input

DBInternalMemoryAddress	-	source base address in internal memory (16-bit)
DBRecordNumber	-	Record number (16-bit)
DBLengthLo	-	number of bytes to write (8-bit)

Output None

Destroys None

Example

```
; Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE

; specify the parameters to write a data array to a random variable record
ld  HL, #MyRecordDataBaseAddress
ld  [DBInternalMemoryAddress], HL
ld  HL, [MyCurrentRecordNumber]
ld  [DBRecordNumber], HL
ld  A, #MyRecordDataSize
ld  [DBLengthLo], A
DB_WRITE_RECORD_RANDOMVAR
```

DB_WRITE_RECORD_WITHOFFSET_RANDOMVAR

Description Writes to a specific record in a variable-sized random access database in external memory starting at a specified offset from the start address of record.

Usage DB_WRITE_RECORD_WITHOFFSET_RANDOMVAR

Assumptions The database file has been previously opened.
The database structure must adhere to the header requirements of a random variable size database structure.
Record size is limited to 256 bytes.

Input

DBInternalMemoryAddress	-	source base address in internal memory (16-bit)
DBRecordNumber	-	Record number (16-bit)
DBLengthLo	-	number of bytes to write (8-bit)
DBRecordOffset	-	offset from start address of record to begin writing (8-bit)

Output None

Destroys None

Example

```
; Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE

; specify the parameters to write a data array to a random variable record
```

```
ld  HL, #MyRecordDataBaseAddress
ld  [DBInternalMemoryAddress], HL
ld  HL, [MyCurrentRecordNumber]
ld  [DBRecordNumber], HL
ld  A, #MyRecordDataSize
ld  [DBLengthLo], A

; specify the offset into the record
ld  A, #MYRECORDDATAFIELD5OFFSET
ld  [DBRecordOffset], A

DB_WRITE_RECORD_WITHOFFSET_RANDOMVAR
```

DB_READ_RECORD

Description Reads a specified record data to database in external memory starting at the start address of record. This is used for reading data from a sequential access and linked list database structure.

Usage DB_READ_RECORD

Assumptions The database file has been previously opened.
Record size is limited to 256 bytes.

Input

DBInternalMemoryAddress	-	destination base address in internal memory (16-bit)
DBExternalMemoryAddress	-	source record offset in external memory (16-bit)
DBLengthLo	-	number of bytes to read (8-bit)

Output None

Destroys None

Example

```
;Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; read in data from eeprom to internal memory
; this example is accessing a sequential type structure
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; specify the base offset of a record in the database
; always 0 for sequential access memory
ld  HL, #0000H
ld  [DBExternalMemoryAddress], HL

; specify the memory buffer to store the data read from eeprom
ld  HL, #MyRecordDataBaseAddress
ld  [DBInternalMemoryAddress], HL

; specify the number of bytes to transfer
ld  A, #MyRecordDataSize
ld  [DBLengthLo], A

; read the data array from eeprom database
```

```
DB_READ_RECORD
```

DB_READ_RECORD_WITHOFFSET

Description Reads a specified record data to database in external memory starting at an offset from the start address of record. This is used for reading data from a sequential access and linked list database structure.

Usage DB_READ_RECORD_WITHOFFSET

Assumptions The database file has been previously opened.
Record size is limited to 256 bytes.

Input

DBInternalMemoryAddress	-	destination base address in internal memory (16-bit)
DBExternalMemoryAddress	-	source record offset in external memory (16-bit)
DBLengthLo	-	number of bytes to read (8-bit)
DBRecordOffset	-	offset from start address of record to begin reading (8-bit)

Output None

Destroys None

Example

```
;Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; read in data from eeprom to internal memory
; this example is accessing a sequential type structure
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; specify the base offset of a record in the database
; always 0 for sequential access memory
ld  HL, #0000H
ld  [DBExternalMemoryAddress], HL

; specify the memory buffer to store the data read from eeprom
ld  HL, #MyRecordDataBaseAddress
ld  [DBInternalMemoryAddress], HL

; specify the number of bytes to transfer
ld  A, #MyRecordDataSize
ld  [DBLengthLo], A

; specify the offset from start of record
ld  A, #MyRecordDataOffsetForField3
ld  [DBRecordOffset], A

; read the data array from eeprom database
DB_READ_RECORD_WITHOFFSET
```

DB_READ_RECORD_RANDOMFIX

Description Read from a specific record in a fixed-sized random access database in external memory starting at the start address of record.

Usage DB_READ_RECORD_RANDOMFIX

Assumptions The database file has been previously opened.
The database structure must adhere to the header requirements of a random fixed size database structure.
Record size is limited to 256 bytes.

Input

DBInternalMemoryAddress	-	destination base address in internal memory (16-bit)
DBRecordNumber	-	Record number (16-bit)
DBLengthLo	-	number of bytes to read (8-bit)

Output None

Destroys None

Example

```
; Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE

;get the absolute eeprom address of the specified record
ld  HL, #MyRecordDataBaseAddress
ld  [DBInternalMemoryAddress], HL
ld  HL, [MyCurrentRecordNumber]
ld  [DBRecordNumber], HL
ld  A, #MyRecordDataSize
ld  [DBLengthLo], A
DB_READ_RECORD_RANDOMFIX


; Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE

; specify the parameters to read from a random fix record to a data array
ld  HL, #MyRecordDataBaseAddress
ld  [DBInternalMemoryAddress], HL
ld  HL, [MyCurrentRecordNumber]
ld  [DBRecordNumber], HL
ld  A, #MyRecordDataSize
ld  [DBLengthLo], A

; read the data from eeprom
DB_READ_RECORD_RANDOMFIX
```

DB_READ_RECORD_WITHOFFSET_RANDOMFIX

Description Reads a specific record in a fixed-sized random access database in external memory starting at a specified offset from the start address of record.

Usage DB_READ_RECORD_WITHOFFSET_RANDOMFIX

Assumptions The database file has been previously opened.
The database structure must adhere to the header requirements of a random fixed size database structure.
Record size is limited to 256 bytes.

Input

DBInternalMemoryAddress	-	destination base address in internal memory (16-bit)
DBRecordNumber	-	Record number (16-bit)
DBLengthLo	-	number of bytes to read (8-bit)
DBRecordOffset	-	offset from start address of record to begin reading (8-bit)

Output None

Destroys None

Example

```
; Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE

;get the absolute eeprom address of the specified record
ld  HL, #MyRecordDataBaseAddress
ld  [DBInternalMemoryAddress], HL
ld  HL, [MyCurrentRecordNumber]
ld  [DBRecordNumber], HL
ld  A, #MyRecordDataSize
ld  [DBLengthLo], A
DB_READ_RECORD_RANDOMFIX


; Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE

; specify the parameters to read from a random fix record to a data array
ld  HL, #MyRecordDataBaseAddress
ld  [DBInternalMemoryAddress], HL
ld  HL, [MyCurrentRecordNumber]
ld  [DBRecordNumber], HL
ld  A, #MyRecordDataSize
ld  [DBLengthLo], A

; specify the offset into the record
ld  A, #MYRECORDDATAFIELD5OFFSET
ld  [DBRecordOffset], A

; read the data from eeprom
DB_READ_RECORD_WITHOFFSET_RANDOMFIX
```

DB_READ_RECORD_RANDOMVAR

Description Read from a specific record in a fixed-sized random access database in external memory starting at the start address of record.

Usage DB_READ_RECORD_RANDOMVAR

Assumptions The database file has been previously opened.
The database structure must adhere to the header requirements of a random variable size database structure.
Record size is limited to 256 bytes.

Input

DBInternalMemoryAddress	-	destination base address in internal memory (16-bit)
DBRecordNumber	-	Record number (16-bit)
DBLengthLo	-	number of bytes to read (8-bit)

Output None

Destroys None

Example

```
; Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE

; specify the parameters to read from random variable size record
ld  HL, #MyRecordDataBaseAddress
ld  [DBInternalMemoryAddress], HL
ld  HL, [MyCurrentRecordNumber]
ld  [DBRecordNumber], HL
ld  A, #MyRecordDataSize
ld  [DBLengthLo], A

;read the record into internal memory
DB_READ_RECORD_WITHOFFSET_RANDOMVAR
```

DB_READ_RECORD_WITHOFFSET_RANDOMVAR

Description Reads a specific record in a variable-sized random access database in external memory starting at a specified offset from the start address of record.

Usage DB_READ_RECORD_WITHOFFSET_RANDOMVAR

Assumptions The database file has been previously opened.
The database structure must adhere to the header requirements of a random variable size database structure.
Record size is limited to 256 bytes.

Input

DBInternalMemoryAddress	-	destination base address in internal memory (16-bit)
DBRecordNumber	-	Record number (16-bit)

	DBLengthLo	-	number of bytes to read (8-bit)
	DBRecordOffset	-	offset from start address of record to begin reading (8-bit)

Output None

Destroys None

Example

```
; Open the associated database of the current application
ld  HL, [CORECurrentADDAddress]
ld  [DBExternalMemoryAddress], HL
DB_OPEN_FILE

; specify the parameters to read from random variable size record
ld  HL, #MyRecordDataBaseAddress
ld  [DBInternalMemoryAddress], HL
ld  HL, [MyCurrentRecordNumber]
ld  [DBRecordNumber], HL
ld  A, #MyRecordDataSize
ld  [DBLengthLo], A

; specify the offset into the record
ld  A, #MYRECORDDATAFIELD5OFFSET
ld  [DBRecordOffset], A

;read the record into internal memory
DB_READ_RECORD_WITHOFFSET_RANDOMVAR
```

Linked-List Operations

DB_REMOVE_RECORD_LINKLIST

Description Removes a record from a link list database structure.

Usage DB_REMOVE_RECORD_LINKLIST

Assumptions Double Linked list database has been previously opened.

Input	HL	-	Absolute Address (internal memory) Link List Head Pointer
	DBRecordAddress	-	Address offset of record to remove from the link list structure
Output	*HL	-	Updated with the new record address offset if removing first record in list.

Destroys None

Example

```
ld  HL, #MyLinkListHeadPointer
ld  BA, [MyCurrentRecordPointer]
DB_REMOVE_RECORD_LINKLIST
```

DB_INSERT_RECORD_LINKLIST

Description Inserts a record into a link list after a specified record.

Usage DB_INSERT_RECORD_LINKLIST

Assumptions Double Linked list database has been previously opened.

Input

HL	-	Absolute Address (internal memory) Link List Head Pointer
DBRecordAddress	-	Address offset of record to remove from the link list structure
DBInsertRecordAddress	-	Address offset of record where the new record is to be appended. If DBNULLRECORDOFFSET, then new record will be inserted as the first record in list.

Output

*HL	-	Updated with the new record address offset if the new record is the first record in list.

Destroys None

Example

```
ld  HL, #MyLinkListHeadPointer
ld  BA, [MyCurrentRecordPointer]
ld  [DBRecordAddress], BA
ld  BA, [MyNewRecordPointer]
ld  [DBInsertRecordAddress], BA
DB_REMOVE_RECORD_LINKLIST
```

DB_LOCATE_INSERTION_BYSIZE_LINKLIST

Description Locates the record where a new record can be inserted. The new record will be inserted to sort the list database according to size of the record. The new record is inserted after the specified record.

Usage DB_LOCATE_INSERTION_BYSIZE_LINKLIST

Assumptions Double Linked list database has been previously opened.

Input

HL	-	record offset to begin searching for a record
DBRecordSize	-	search criteria to locate where to insert a record in the list

Output

BA	-	Results of the search function:
		
		`BAReg != DBNULLRECORDOFFSET`
		Record address offset where a new record can be inserted after.
		
		`BAReg == DBNULLRECORDOFFSET`
		Record should be located as the first record in the list.

Destroys None

Example

```
; locate the address where we can insert the new record
```

```
ld  HL, #MyUnusedLinkListHeadPointer
ld  A, #SizeOfRecordToSearch
ld  [DBRecordSize], A
DB_LOCATE_INDERTION_BYSIZE_LINKLIST

; insert the new record at the address return by the previous operation
ld  [DBInsertRecordAddress], BA
ld  HL, #MyLinkListHeadPointer
ld  BA, [MyCurrentRecordPointer]
ld  [DBRecordAddress], BA
DB_REMOVE_RECORD_LINKLIST
```

DISPLAY API

These are used to handle writing characters, numbers, and messages to the display device.

Starting Segment Digit Positions

Constant
LCDSEGDIGIT1
LCDSEGDIGIT2
LCDSEGDIGIT3
LCDSEGDIGIT4
LCDSEGDIGIT5
LCDSEGDIGIT6

Starting Main Dot-Matrix Line 1 Digit Positions:

Constant
LCDMAINDMLINE1COL1
LCDMAINDMLINE1COL2
LCDMAINDMLINE1COL3
LCDMAINDMLINE1COL4
LCDMAINDMLINE1COL5
LCDMAINDMLINE1COL6
LCDMAINDMLINE1COL7
LCDMAINDMLINE1COL8
LCDMAINDMLINE1COL9
LCDMAINDMLINE1COL10
LCDMAINDMLINE1COL11
LCDMAINDMLINE1COL12
LCDMAINDMLINE1COL13
LCDMAINDMLINE1COL14
LCDMAINDMLINE1COL15
LCDMAINDMLINE1COL16
LCDMAINDMLINE1COL17
LCDMAINDMLINE1COL18
LCDMAINDMLINE1COL19
LCDMAINDMLINE1COL20
LCDMAINDMLINE1COL21
LCDMAINDMLINE1COL22
LCDMAINDMLINE1COL23
LCDMAINDMLINE1COL24
LCDMAINDMLINE1COL25
LCDMAINDMLINE1COL26
LCDMAINDMLINE1COL27
LCDMAINDMLINE1COL28
LCDMAINDMLINE1COL29

LCDMAINDMLINE1COL30
LCDMAINDMLINE1COL31
LCDMAINDMLINE1COL32
LCDMAINDMLINE1COL33
LCDMAINDMLINE1COL34
LCDMAINDMLINE1COL35
LCDMAINDMLINE1COL36
LCDMAINDMLINE1COL37
LCDMAINDMLINE1COL38
LCDMAINDMLINE1COL39
LCDMAINDMLINE1COL40
LCDMAINDMLINE1COL41
LCDMAINDMLINE1COL42
LCDMAINDMLINE1COL43

Starting Main Dot-Matrix Line 2 Digit Positions:

Constant
LCDMAINDMLINE2COL1
LCDMAINDMLINE2COL2
LCDMAINDMLINE2COL3
LCDMAINDMLINE2COL4
LCDMAINDMLINE2COL5
LCDMAINDMLINE2COL6
LCDMAINDMLINE2COL7
LCDMAINDMLINE2COL8
LCDMAINDMLINE2COL9
LCDMAINDMLINE2COL10
LCDMAINDMLINE2COL11
LCDMAINDMLINE2COL12
LCDMAINDMLINE2COL13
LCDMAINDMLINE2COL14
LCDMAINDMLINE2COL15
LCDMAINDMLINE2COL16
LCDMAINDMLINE2COL17
LCDMAINDMLINE2COL18
LCDMAINDMLINE2COL19
LCDMAINDMLINE2COL20
LCDMAINDMLINE2COL21
LCDMAINDMLINE2COL22
LCDMAINDMLINE2COL23
LCDMAINDMLINE2COL24
LCDMAINDMLINE2COL25
LCDMAINDMLINE2COL26
LCDMAINDMLINE2COL27
LCDMAINDMLINE2COL28

LCDMAINDMLINE2COL29
LCDMAINDMLINE2COL30
LCDMAINDMLINE2COL31
LCDMAINDMLINE2COL32
LCDMAINDMLINE2COL33
LCDMAINDMLINE2COL34
LCDMAINDMLINE2COL35
LCDMAINDMLINE2COL36
LCDMAINDMLINE2COL37
LCDMAINDMLINE2COL38
LCDMAINDMLINE2COL39
LCDMAINDMLINE2COL40
LCDMAINDMLINE2COL41
LCDMAINDMLINE2COL42
LCDMAINDMLINE2COL43

Starting Upper Dot-Matrix Digit Positions:

Constant
LCDUPPERDMCOL1
LCDUPPERDMCOL2
LCDUPPERDMCOL3
LCDUPPERDMCOL4
LCDUPPERDMCOL5
LCDUPPERDMCOL6
LCDUPPERDMCOL7
LCDUPPERDMCOL8
LCDUPPERDMCOL9
LCDUPPERDMCOL10
LCDUPPERDMCOL11
LCDUPPERDMCOL12

Large Font Digit Positions

Constant
LCDBIGCHARDMCOL1
LCDBIGCHARDMCOL2
LCDBIGCHARDMCOL3
LCDBIGCHARDMCOL4
LCDBIGCHARDMCOL5
LCDBIGCHARDMCOL6
LCDBIGCHARDMCOL7
LCDBIGCHARDMCOL8
LCDBIGCHARDMCOL9
LCDBIGCHARDMCOL10
LCDBIGCHARDMCOL11

```
LCDBIGCHARDMCOL12
LCDBIGCHARDMCOL13
LCDBIGCHARDMCOL14
LCDBIGCHARDMCOL15
LCDBIGCHARDMCOL16
LCDBIGCHARDMCOL17
LCDBIGCHARDMCOL18
LCDBIGCHARDMCOL19
LCDBIGCHARDMCOL20
LCDBIGCHARDMCOL21
LCDBIGCHARDMCOL22
LCDBIGCHARDMCOL23
LCDBIGCHARDMCOL24
LCDBIGCHARDMCOL25
LCDBIGCHARDMCOL26
LCDBIGCHARDMCOL27
LCDBIGCHARDMCOL28
LCDBIGCHARDMCOL29
LCDBIGCHARDMCOL30
LCDBIGCHARDMCOL31
LCDBIGCHARDMCOL32
LCDBIGCHARDMCOL33
LCDBIGCHARDMCOL34
LCDBIGCHARDMCOL35
LCDBIGCHARDMCOL36
LCDBIGCHARDMCOL37
LCDBIGCHARDMCOL38
LCDBIGCHARDMCOL39
LCDBIGCHARDMCOL40
LCDBIGCHARDMCOL41
LCDBIGCHARDMCOL42
LCDBIGCHARDMCOL43
```

Flag Address and Bit Definitions

```
L_ADDR                  bLCDBITPATTERN_L
A_ADDR                  bLCDBITPATTERN_A
P_ADDR                  bLCDBITPATTERN_P
TAIL_ADDR               bLCDBITPATTERN_TAIL
TMR_ADDR                bLCDBITPATTERN_TMR
ARROW_ADDR              bLCDBITPATTERN_ARROW
CHR_ADDR                bLCDBITPATTERN_CHR
MOON_ADDR               bLCDBITPATTERN_MOON
ACLK_ADDR               bLCDBITPATTERN_ACLK
NOTE_ADDR               bLCDBITPATTERN_NOTE
PER1_ADDR               bLCDBITPATTERN_PER1
```

```
DASH1_ADDR              bLCDBITPATTERN_DASH1
PER2_ADDR               bLCDBITPATTERN_PER2
DASH2_ADDR              bLCDBITPATTERN_DASH2
```

9-Segment Character Set

```
SEG_0                   equ     0
SEG_1                   equ     1
SEG_2                   equ     2
SEG_3                   equ     3
SEG_4                   equ     4
SEG_5                   equ     5
SEG_6                   equ     6
SEG_7                   equ     7
SEG_8                   equ     8
SEG_9                   equ     9
SEG_SPACE               equ     10
SEG_A                   equ     11
SEG_B                   equ     12
SEG_C                   equ     13
SEG_D                   equ     14
SEG_E                   equ     15
SEG_F                   equ     16
SEG_G                   equ     17
SEG_H                   equ     18
SEG_I                   equ     19
SEG_J                   equ     20
SEG_K                   equ     21
SEG_L                   equ     22
SEG_M                   equ     23
SEG_N                   equ     24
SEG_O                   equ     SEG_0
SEG_P                   equ     25
SEG_Q                   equ     26
SEG_R                   equ     27
SEG_S                   equ     SEG_5
SEG_T                   equ     28
SEG_U                   equ     29
SEG_V                   equ     30
SEG_W                   equ     31
SEG_Y                   equ     32
SEG_Z                   equ     SEG_2

SEG_MINUS               equ     33
SEG_DASH                equ     SEG_MINUS
SEG_PLUS                equ     34
```

```
SEG_COLON               equ     SEG_I
SEG_OPENPAR             equ     SEG_C
SEG_CLOSEPAR            equ     35
SEG_DOLLAR              equ     36
```

5x5 Dot Matrix Character Set

```
DM5_0                   equ     0
DM5_1                   equ     1
DM5_2                   equ     2
DM5_3                   equ     3
DM5_4                   equ     4
DM5_5                   equ     5
DM5_6                   equ     6
DM5_7                   equ     7
DM5_8                   equ     8
DM5_9                   equ     9
DM5_SPACE               equ     10
DM5_A                   equ     11
DM5_B                   equ     12
DM5_C                   equ     13
DM5_D                   equ     14
DM5_E                   equ     15
DM5_F                   equ     16
DM5_G                   equ     17
DM5_H                   equ     18
DM5_I                   equ     19
DM5_J                   equ     20
DM5_K                   equ     21
DM5_L                   equ     22
DM5_M                   equ     23
DM5_N                   equ     24
DM5_O                   equ     25
DM5_P                   equ     26
DM5_Q                   equ     27
DM5_R                   equ     28
DM5_S                   equ     29
DM5_T                   equ     30
DM5_U                   equ     31
DM5_V                   equ     32
DM5_W                   equ     33
DM5_X                   equ     34
DM5_Y                   equ     35
DM5_Z                   equ     36

DM5_EXCLAMATION         equ     37
DM5_DBLQUOTE            equ     38
```

```
DM5_NUMBER              equ     39
DM5_DOLLAR              equ     40
DM5_PERCENT             equ     41
DM5_AMPERSAND           equ     42
DM5_SGLQUOTE            equ     43
DM5_OPENPAR             equ     44
DM5_CLOSEPAR            equ     45
DM5_ASTERISK            equ     46
DM5_PLUS                equ     47
DM5_COMMA               equ     48
DM5_DASH                equ     49
DM5_MINUS               equ     DM5_DASH
DM5_PERIOD              equ     50
DM5_SLASH               equ     51
DM5_COLON               equ     52
DM5_SEMICOLON           equ     53
DM5_LESSTHAN            equ     54
DM5_EQUAL               equ     55
DM5_GREATERTHAN         equ     56
DM5_QUESTION            equ     57
DM5_ATREVERSED          equ     58
DM5_OPENSQBRACKET       equ     59
DM5_BACKSLASH           equ     60
DM5_CLOSESQBRACKET      equ     61
DM5_CIRCUMFLEX          equ     62
DM5_UNDERSCORE          equ     63
DM5_BACKAPOSTROPHE      equ     64
DM5_OPENBRACE           equ     65
DM5_VERTBAR             equ     66
DM5_CLOSEBRACE          equ     67
DM5_TILDE               equ     68
DM5_SECTION             equ     69

DM5_EURO                equ     70
DM5_POUND               equ     71
DM5_YEN                 equ     72
DM5_AUMLAUT             equ     73
DM5_ARING               equ     74
DM5_AELIGATURE          equ     75
DM5_CCEDILLA            equ     76
DM5_NTILDE              equ     77
DM5_OUMLAUT             equ     78
DM5_OSLASH              equ     79
DM5_UUMLAUT             equ     80
DM5_SZLIGATURE          equ     81
DM5_INVEXCLAMATION      equ     82
DM5_INVQUESTION         equ     83
```

```
DM5_FEMORDINAL          equ     84
DM5_DEGREE              equ     85
DM5_MACRON              equ     86

DM5_SPADE               equ     87
DM5_CLUB                equ     88
DM5_HEART               equ     89
DM5_DIAMOND             equ     90
DM5_TEN                 equ     91
DM5_NEWMOON             equ     92
DM5_FIRSTQUARTER        equ     93
DM5_LASTQUARTER         equ     94
DM5_DOWNARROW           equ     95
DM5_UPARROW             equ     96
DM5_AM                  equ     97
DM5_PM                  equ     98
DM5_MFORAMPM            equ     99
DM5_COMPRESS_1          equ     100
DM5_LEFTARROW           equ     101
DM5_RIGHTARROW          equ     102
DM5_CURSOR              equ     103
DM5_SENTINEL            equ     104
DM5_BLANK               equ     105
```

8x8 Dot Matrix Character Set

```
DM8_0                   equ     0
DM8_1                   equ     1
DM8_2                   equ     2
DM8_3                   equ     3
DM8_4                   equ     4
DM8_5                   equ     5
DM8_6                   equ     6
DM8_7                   equ     7
DM8_8                   equ     8
DM8_9                   equ     9
DM8_SPACE               equ     10
DM8_A                   equ     11
DM8_L                   equ     12
DM8_P                   equ     13
DM8_S                   equ     14
DM8_T                   equ     15
```

8x4 Dot Matrix Character Set

```
DM8_I                   equ     16
```

8x1 Dot Matrix Character Set

```
DM8_PERIOD          equ     17
DM8_DOT             equ     17
DM8_COLON           equ     18
DM8_DASH            equ     19
```

Character Bitmap Patterns

The table below shows the font for the nine-segment character positions of the LCD. Some letters cannot be reasonably represented with nine segments, and they are shown as blank characters in the table.

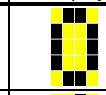
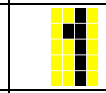
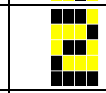
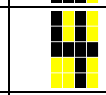
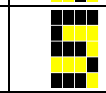
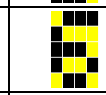
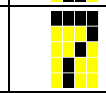
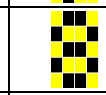
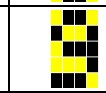
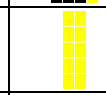
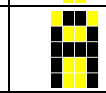
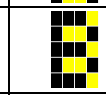
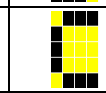
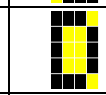
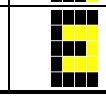
Segment Display Patterns

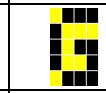
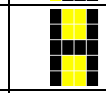
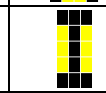
No.	Character	Display
0	0	
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
10	A	
11	B	
12	C	
13	D	
14	E	
15	F	

No.	Character	Display
16	G	
17	H	
18	I	
19	J	
20	K	
21	L	
22	M	
23	N	
24	O	
25	P	
26	Q	
27	R	
28	S	
29	T	
30	U	
31	V	

No.	Character	Display
32	W	
33	X	
34	Y	
35	Z	
36	space	
37	- dash	
38	+	
39	:	
40	(	
41	)	
42	$	

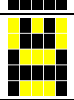
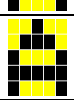
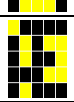
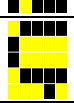
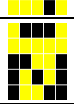
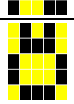
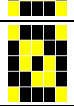
Small Font Dot-Matrix Patterns

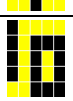
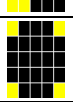
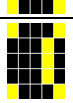
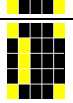
No.	Character	Fixed-Width Display	Proportional Display
0	0		
1	1		
2	2		
3	3		
4	4		
5	5		
6	6		
7	7		
8	8		
9	9		
10	blank		
11	A		
12	B		
13	C		
14	D		
15	E		

No.	Character	Fixed-Width Display	Proportional Display
16	F		
17	G		
18	H		
19	I		
20	J		
21	K		
22	L		
23	M		
24	N		
25	O		
26	P		
27	Q		
28	R		
29	S		
30	T		
31	U		

No.	Character	Fixed-Width Display	Proportional Display
32	V		
33	W		
34	X		
35	Y		
36	Z		
37	!		
38	“		
39	#		
40	$		
41	%		
42	&		
43	‘		
44	(		
45	)		
46	*		
47	+		

No.	Character	Fixed-Width Display	Proportional Display
48	,		
49	- dash		
50	.		
51	/		
52	:		
53	;		
54	<		
55	=		
56	>		
57	?		
58	@ reversed (US, Span, Port)		
59	[		
60	\		
61	]		
62	^		
63	_ underscore		

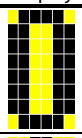
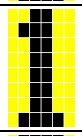
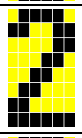
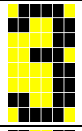
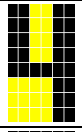
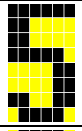
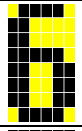
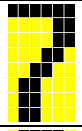
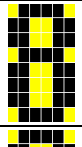
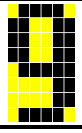
No.	Character	Fixed-Width Display	Proportional Display
64	`		
65	{		
66	|		
67	}		
68	~		
69	§		
70	euro		
71	£		
72	¥ yen		
73	Ä		
74	Å		
75	Æ		
76	Ç		
77	Ñ		
78	Ö		
79	Ø		

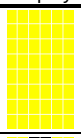
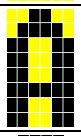
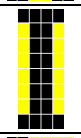
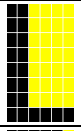
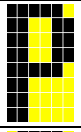
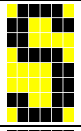
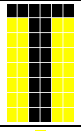
No.	Character	Fixed-Width Display	Proportional Display
80	Ü		
81	ß		
82	¡		
83	¿		
84	ª		
85	º		
86	¯		
87	? spade		
88	? club		
89	? heart		
90	? diamond		
91	10		
92	new moon		
93	first quarter		
94	last quarter		
95	↓ down arrow		

No.	Character	Fixed-Width Display	Proportional Display
96	↑ up arrow		
97	A AM indicator		
98	P PM indicator		
99	M AM/PM indicator		
100	1 compressed 1		
101	? left arrow		
102	? right arrow		
103	¦ cursor		
104	¦ sentinel		

No.	Character	Fixed-Width Display	Proportional Display

Large Font Dot-Matrix Patterns

No.	Character	Proportional Display
0	0	
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	

No.	Character	Proportional Display
10	blank	
11	A	
12	I	
13	L	
14	P	
15	S	
16	T	
17	.	
18	:	
19	- dash	

Flag Display

LCD_CLR_ALL_FLAGS

Description Clear all the LCD flags.

Usage LCD_CLR_ALL_FLAGS

Assumptions None

Input None

Output None

Destroys BA, IX, HL

Example
```
; Turn off all flag Icons on LCD display
LCD_CLR_ALL_FLAGS
```

LCD_UPD_L_FLAG

Description Clears or displays the “L” icon.

Usage LCD_UPD_L_FLAG { ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Clear L flag, but display A flag
LCD_UPD_L_FLAG OFF
LCD_UPD_A_FLAG ON
```

LCD_UPD_A_FLAG

Description Clears or displays the “A” icon.

Usage LCD_UPD_A_FLAG { ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Clear L flag, but display A flag
LCD_UPD_L_FLAG OFF
LCD_UPD_A_FLAG ON
```

LCD_UPD_P_FLAG

Description Clears or displays the “P” icon.

Usage LCD_UPD_P_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Clear L flag and P flags, but display A flag
LCD_UPD_L_FLAG OFF
LCD_UPD_A_FLAG ON
LCD_UPD_P_FLAG OFF
```

LCD_UPD_LAP_FLAG

Description Clears or displays the “LAP” an icon (a combination of the L, A, and P flags)

Usage LCD_UPD_LAP_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
;display the LAP icon
LCD_UPD_LAP_FLAG ON
```

LCD_UPD_AP_FLAG

Description Clears or displays the “AP” icon combination of A/P flags.

Usage LCD_UPD_AP_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
;Clear the AP flag, and display the L flag
LCD_UPD_AP_FLAG OFF
LCD_UPD_L_FLAG ON
```

LCD_UPD_NOTE_FLAG

Description Clears or displays the “NOTE” icon.

Usage LCD_UPD_NOTE_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Display the NOTE flag
LCD_UPD_NOTE_FLAG ON
```

LCD_UPD_ACLK_FLAG

Description Clears or displays the “ALARM CLOCK” icon.

Usage LCD_UPD_ACLK_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Clear the ALARM CLOCK flag
LCD_UPD_ACLK_FLAG  OFF
```

LCD_UPD_TAIL_FLAG

Description Clears or displays the “TAIL” icon.

Usage LCD_UPD_TAIL_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Display the TAIL and the TIMER to indicate CDR function
LCD_UPD_TAIL_FLAG ON
LCD_UPD_TMR_FLAG ON
```

LCD_UPD_ARROW_FLAG

Description Clears or displays the “ARROW” icon.

Usage LCD_UPD_ARROW_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Clear the ARROW and TAIL, and display the TIMER to show CDS function
LCD_UPD_ARROW_FLAG OFF
LCD_UPD_TAIL _FLAG OFF
```

LCD_UPD_TMR_FLAG

Description Clears or displays the “TIMER” icon.

Usage LCD_UPD_TMR_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Clear the TIMER flag to indicate the TIMER has stopped
LCD_UPD_TMR_FLAG OFF
```

LCD_UPD_CDC_FLAG

Description Clears or displays the “COUNT DOWN TO CHRONO” icon combination, which consists of the TIMER/ARROW/STOPWATCH flags.

Usage LCD_UPD_CDC_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Display CDC flag to indicate CDC function is active.
LCD_UPD_CDC_FLAG ON
```

LCD_UPD_CDR_FLAG

Description Clears or displays the “COUNT DOWN AND REPEAT” icon combination of TIMER/TAIL flags.

Usage LCD_UPD_CDR_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Display CDR flag to indicate CDR function is active.
LCD_UPD_CDR_FLAG ON
```

LCD_UPD_CHR_FLAG

Description Clears or displays the “STOPWATCH” icon.

Usage LCD_UPD_CHR_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Display CHR flag to indicate CHRONO is active.
LCD_UPD_CHR_FLAG ON
```

LCD_UPD_MOON_FLAG

Description Clears or displays the “MOON” icon.

Usage LCD_UPD_MOON_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Clear the MOON flag to indicate NightMode is disabled.
LCD_UPD_MOON_FLAG  OFF
```

LCD_UPD_DASH1_FLAG

Description Clears or displays the “DASH 1” icon.

Usage LCD_UPD_DASH1_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Turn on the DASH between Segmented Digits 3 and 4
LCD_UPD_DASH1_FLAG ON
```

LCD_UPD_PER1_FLAG

Description Clears or displays the “PERIOD 1” icon.

Usage LCD_UPD_PER1_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Turn on the PERIOD between Segmented Digits 3 and 4
LCD_UPD_PER1_FLAG ON
```

LCD_UPD_COLON1_FLAG

Description Clears or displays the “COLON 1” icon.

Usage LCD_UPD_COLON1_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Turn on the COLON between Segmented Digits 3 and 4
LCD_UPD_ COLON1_FLAG ON
```

LCD_UPD_DASH2_FLAG

Description Clears or displays the “DASH 2” icon.

Usage LCD_UPD_DASH2_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Turn on the DASH between Segmented Digits 4 and 5
LCD_UPD_ DASH2_FLAG ON
```

LCD_UPD_PER2_FLAG

Description Clears or displays the “PERIOD 2” icon.

Usage LCD_UPD_PER2_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Turn on the PERIOD between Segmented Digits 4 and 5
LCD_UPD_ PER2_FLAG ON
```

LCD_UPD_COLON2_FLAG

Description Clears or displays the “COLON 2” icon.

Usage LCD_UPD_COLON2_FLAG{ ON | OFF }

Assumptions None

Input None

Output None

Destroys HL

Example
```
; Turn on the COLON between Segmented Digits 4 and 5
LCD_UPD_COLON2_FLAG ON
```

Clearing Display

LCD_CLR_2DIGIT_SEG

Description Clears two 9-segment character positions.

Usage LCD_CLR_2DIGIT_SEG

Assumptions None

Input IX - Leftmost starting digit position we wish to clear.

Output None

Destroys A, B, L, IX

Example

```
; CLear digits 3 and 4 in the segmented display area.
ld   IX, #LCDSEGDIGIT3
LCD_CLR_2DIGIT_SEG
```

LCD_CLR_3DIGIT_SEG

Description Clears three consecutive 9-segment character positions

Usage LCD_CLR_3DIGIT_SEG

Assumptions None

Input IX - Leftmost starting digit position we wish to clear.

Output None

Destroys A, B, L, IX

Example

```
; CLear digits 1,2 and 3 in the segmented display area.
ld   IX, #LCDSEGDIGIT1
LCD_CLR_3DIGIT_SEG
```

LCD_CLR_SMALL_PROP_WIDTH_2DIG_DM_DATA

Description Clear a small-font, proportional-width 2-digit data.

Usage LCD_CLR_SMALL_PROP_WIDTH_2DIG_DM_DATA

Assumptions None

Input IX - Starting dot matrix column and line position.

Output None

Destroys A, B, L, IX

Example

```
; Clear 2 digit space starting at column 20, in line 2 of the big dot
; matrix
ld   IX, #LCDMAINDMLINE2COL20
LCD_CLR_SMALL_PROP_WIDTH_2DIG_DM_DATA
```

LCD_CLR_SMALL_FIXED_WIDTH_2DIG_DM_DATA

Description Clear a small-font, fixed-width 2-digit data.

Usage LCD_CLR_SMALL_FIXED_WIDTH_2DIG_DM_DATA

Assumptions None

Input IX - Starting dot matrix column and line position

Output None

Destroys A, B, L, IX

Example

```
; Clear 2 digit space starting at column 20, in line 2 of the big dot
; matrix
ld   IX, #LCDMAINDMLINE2COL20
LCD_CLR_SMALL_FIXED_WIDTH_2DIG_DM_DATA
```

LCD_CLR_SMALL_PROP_WIDTH_3DIG_DM_DATA

Description Clear a small-font, proportional-width 3-digit data.

Usage LCD_CLR_SMALL_PROP_WIDTH_3DIG_DM_DATA

Assumptions None

Input IX - Starting dot matrix column and line position

Output None

Destroys A, B, L, IX

Example
```
; Clear 3 digit space starting at column 22, in line 1 of the big dot
matrix
ld   IX, #LCDMAINDMLINE1COL22
LCD_CLR_SMALL_PROP_WIDTH_3DIG_DM_DATA
```

LCD_CLR_SMALL_FIXED_WIDTH_3DIG_DM_DATA

Description Clear a small-font, fixed-width 3-digit data.

Usage LCD_CLR_SMALL_FIXED_WIDTH_3DIG_DM_DATA

Assumptions None

Input IX - Starting dot matrix column and line position

Output None

Destroys A, B, L, IX

Example
```
; Clear 3 digit space starting at column 22, in line 1 of the big dot
matrix
ld   IX, #LCDMAINDMLINE1COL22
LCD_CLR_SMALL_FIXED_WIDTH_3DIG_DM_DATA
```

Clearing Large-Font Digits

LCD_CLR_BIG_2DIGIT_DM_DATA

Description Clears a large-font, 2-digit DM data.

Usage LCD_CLR_BIG_2DIGIT_DM_DATA

Assumptions None

Input IX - Starting dot matrix column position

Output None

Destroys A, B, L, IX

Example
```
; Clear 2 digit space starting at column 21, of the big dot matrix
ld   IX, #LCDBIGCHARDMCOL21
LCD_CLR_BIG_2DIGIT_DM_DATA
```

LCD_CLR_BIG_3DIGIT_DM_DATA

Description Clears a large-font, 3-digit DM data.

Usage LCD_CLR_BIG_3DIGIT_DM_DATA

Assumptions None

Input IX - Starting dot matrix column position

Output None

Destroys A, B, L, IX

Example
```
; Clear 3 digit space starting at column 21, of the big dot matrix
ld   IX, #LCDBIGCHARDMCOL21
LCD_CLR_BIG_3DIGIT_DM_DATA
```

Clearing Display Regions

LCD_CLR_DISPLAY

Description Clears the entire LCD.

Usage LCD_CLR_DISPLAY

Assumptions None

Input None

Output None

Destroys BA, HL

Example
```
; Clear the entire display
LCD_CLR_DISPLAY
```

LCD_FILL_DISPLAY

Description Display all segments in the LCD.

Usage LCD_FILL_DISPLAY

Assumptions None

Input None

Output None

Destroys BA, HL

Example
```
; Turn on all segments of the LCD display
LCD_FILL_DISPLAY
```

LCD_CLR_SEG_LINE

Description Clears the entire 9-segment line. This does NOT include the punctuation.

Usage LCD_CLR_SEG_LINE

Assumptions None

Input None

Output None

Destroys A, B, L, IX

Example
```
 ; Clear all 6 9-segment digits, not the punctuation
 LCD_UPD_ PER2_FLAG ON
 LCD_CLR_SEG_LINE
```

LCD_CLEAR_UPPER_DM

Description Clear upper-dot matrix area.

Usage LCD_CLEAR_UPPER_DM

Assumptions None

Input None

Output None

Destroys A, B, IX

Example
```
; Clear the upper dot motrix display
LCD_UPD_ PER2_FLAG ON
```

LCD_CLEAR_MAIN_DM

Description Clear main dot matrix area.

Usage LCD_CLEAR_MAIN_DM

Assumptions None

Input None

Output None

Destroys A, IX

Example
```
; Clear the main dot matrix display
LCD_CLEAR_MAIN_DM
```

LCD_CLR_MAIN_DM_LINE1

Description Clear line 1 of the main dot matrix area.

Usage LCD_CLR_MAIN_DM_LINE1

Assumptions None

Input None

Output None

Destroys A, IX

Example
```
; Clear line 1 ONLY of the main dot matrix display. Line 2 is unaffected.
LCD_CLR_MAIN_DM_LINE1
```

LCD_CLR_MAIN_DM_LINE2

Description Clear line 2 of the main dot matrix area.

Usage LCD_CLR_MAIN_DM_LINE2

Assumptions None

Input None

Output None

Destroys A, IX

Example

```
; Clear line 2 ONLY of the main dot matrix display. Line 1 is unaffected.
LCD_CLR_MAIN_DM_LINE2
```

Data Formatting

LCD_FORMAT_3DIGIT_DATA_SUP_ZERO

Description Format a 3-digit BCD data by deleting all leading zeros.

Usage LCD_FORMAT_3DIGIT_DATA_SUP_ZERO

Assumptions None

Input

B	-	100's digit BCD data
A	-	10's digit BCD data (high nibble) 1's digit BCD data (low nibble)

Output

LCDFormatData+2 LCDFormatData+1 LCDFormatData+0	-	Contain the formatted 3 digit data, from most significant digit to least significant digit.

Destroys BA

Example

```
; Format the BCD value 0x10 for display. This will result in a space
followed
; by “10”
; Set BA = 0x010
ld   B, #0
ld   A, #10H
LCD_FORMAT_3DIGIT_DATA_SUP_ZERO
```

LCD_FORMAT_2DIGIT_DATA_SUP_ZERO

Description Format a 2-digit BCD data by deleting all leading zeros.

Usage LCD_FORMAT_2DIGIT_DATA_SUP_ZERO

Assumptions None

Input

A	-	10's digit BCD data (high nibble) 1's digit BCD data (low nibble)

Output

LCDFormatData+1 LCDFormatData+0	-	Contain the formatted 2 digit data, from most significant digit to least significant digit.

Destroys BA

Example

```
 ; Format the BCD #0x89 for display
ld   A, #89H
LCD_FORMAT_2DIGIT_DATA_SUP_ZERO
```

LCD_FORMAT_3DIGIT_DATA_NO_LSD_SUP

Description Format a 3-digit BCD data by deleting all leading zeros except on the LSD.

Usage LCD_FORMAT_3DIGIT_DATA_NO_LSD_SUP

Assumptions None

Input

B	-	100's digit BCD data
A	-	10's digit BCD data (high nibble) 1's digit BCD data (low nibble)

Output

LCDFormatData+2 LCDFormatData+1 LCDFormatData+0	-	Formatted 3 digit data

Destroys BA

Example

```
; Format the BCD value 0x000 for display. After the macro call, this will
be ; displayed as 2 spaces followed by a “0”
ld   B, #0
ld   A, #0
```

LCD_FORMAT_2DIGIT_DATA_SUP_ZERO_MSD

Description Format a 2-digit BCD data by deleting the MSD if equal to zero.

Usage LCD_FORMAT_2DIGIT_DATA_SUP_ZERO_MSD

Assumptions None

Input

A	-	10's digit BCD data (high nibble) 1's digit BCD data (low nibble)

Output

LCDFormatData+1 LCDFormatData+0	-	Contain the formatted 2 digit data, from most significant digit to least significant digit.

Destroys BA

Example

```
;format 2-digit number with suppression of MSB if zero
ld  A, #05H
LCD_FORMAT_2DIGIT_DATA_SUP_ZERO_MSD

;expected result:
;LCDFormatData[1] = SPACE
;LCDFormatData[0] = 5

;format 2-digit number with suppression of MSB if zero
ld  A, #25H
LCD_FORMAT_2DIGIT_DATA_SUP_ZERO_MSD

;expected result:
;LCDFormatData[1] = 2
;LCDFormatData[0] = 5
```

LCD_FORMAT_3DIGIT_DATA_NO_ZERO_SUP

Description Format a 3-digit BCD data with no zero suppression on all digits.

Usage LCD_FORMAT_3DIGIT_DATA_NO_ZERO_SUP

Assumptions None

Input

B	-	100's digit BCD data
A	-	10's digit BCD data (high nibble) 1's digit BCD data (low nibble)

Output

LCDFormatData+2 LCDFormatData+1 LCDFormatData+0	-	Contain the formatted 3 digit data, from most significant digit to least significant digit.

Destroys BA

Example

```
;format 2-digit number with suppression of MSB if zero
ld  BA, #0123H
LCD_FORMAT_3DIGIT_DATA_NO_ZERO_SUP
```

```
;expected result:
;LCDFormatData[2] = 1
;LCDFormatData[1] = 2
;LCDFormatData[0] = 3
```

LCD_FORMAT_2DIGIT_DATA_NO_ZERO_SUP

Description Format a 2-digit BCD data with no zero suppression on all digits.

Usage LCD_FORMAT_2DIGIT_DATA_NO_ZERO_SUP

Assumptions None

Input

A	-	10's digit BCD data (high nibble) 1's digit BCD data (low nibble)

Output

LCDFormatData+1	-	Contain the formatted 2 digit data, from most significant digit to least significant digit.
LCDFormatData+0		

Destroys BA

Example

```
; Format the BCD value 0x00 for display. After the macro call, this will be
; displayed as “00”
ld   A, #0
LCD_FORMAT_2DIGIT_DATA_NO_ZERO_SUP
```

Character Display

LCD_DISP_SEG_CHAR

Description Display a segmented character.

Usage LCD_DISP_SEG_CHAR

Assumptions None

Input

L	-	character to be displayed from 9-segment character set
IX	-	starting digit position

Output None

Destroys A, B, L

Example

```
; Display the Character “C” in digit 4 of the segmented display.
ld   L, #SEG_C
ld   IX, #LCDSEGDIGIT4
LCD_DISP_SEG_CHAR
```

LCD_DISP_SMALL_PROP_WIDTH_DM_CHAR

Description Display small-font, proportional-width dot matrix character.

Usage LCD_DISP_SMALL_PROP_WIDTH_DM_CHAR

Assumptions None

Input

L	-	character to be displayed from the small font dot-matrix character set.
IX	-	starting DM column

Output

bOverflowFlag	-	1 = DM display address is out of range 0 = DM address is in range (IXReg points to the next column)

Destroys BA, H, IX

Example

```
; Display the Character “C” in column 4, line 2 of the main dot matrix.
ld   L, #DM5_C
ld   IX, #LCDMAINDMLINE2COL4
LCD_DISP_SMALL_PROP_WIDTH_DM_CHAR
```

LCD_DISP_SMALL_FIXED_WIDTH_DM_CHAR

Description Display small-font, fixed-width dot matrix character.

Usage LCD_DISP_SMALL_FIXED_WIDTH_DM_CHAR

Assumptions None

Input

L	-	character to be displayed from the small font dot matrix character set.
IX	-	starting DM column

Output

bOverflowFlag	-	1 = DM display address is out of range 0 = DM address is in range (IXReg points to the next column)

Destroys BA, H, IX

Example

```
; Display the Character “C” in column 5, line 1 of the main dot matrix.
ld   L, #DM5_C
ld   IX, #LCDMAINDMLINE1COL5
LCD_DISP_SMALL_FIXED_WIDTH_DM_CHAR
```

Large-Font Character Display

LCD_DISP_BIG_DM_CHAR

Description Display a large-font, dot matrix character.

Usage LCD_DISP_BIG_DM_CHAR

Assumptions None

Input

L	-	character to be displayed from the large font dot matrix character set.
IX	-	starting DM column

Output

bOverflowFlag	-	1 = DM display address is out of range 0 = DM address is in range (IXReg points to the next column)

Destroys L, IX

Example

```
; Display the large Character “S” in column 25 of the main dot matrix.
ld   L, #DM8_S
ld   IX, #LCDBIGCHARDMCOL25
LCD_DISP_BIG_DM_CHAR
```

Numeric Display

LCD_DISP_2DIG_SEG_DATA_WITH_ZERO_SUP

Description Display a 2-digit BCD data with zero suppression on both digits in the 9-segment area.

Usage LCD_DISP_2DIG_SEG_DATA_WITH_ZERO_SUP

Assumptions None

Input

A	-	packed BCD data to be displayed.
IX	-	Leftmost 9-segment digit position where we wish to display the data

Output None

Destroys A, B, L, IX

Example

```
; Display the BCD value 0x00. After the macro call, this will be
; displayed as 2 spaces
ld   A, #0
ld   IX, #LCDSEGDIGIT1
LCD_DISP_2DIG_SEG_DATA_WITH_ZERO_SUP
```

LCD_DISP_3DIG_SEG_DATA_WITH_ZERO_SUP

Description Display a 3-digit segmented BCD data with zero suppression on leading digits.

Usage LCD_DISP_3DIG_SEG_DATA_WITH_ZERO_SUP

Assumptions None

Input

B	-	100's digit BCD data
A	-	packed 10's and 1's digit data
IX	-	Leftmost 9-segment digit position where we wish to display the data

Output None

Destroys A, B, L, IX

Example

```
; Display the BCD value 0x000 After the macro call, this will be
; displayed as 3 spaces
ld   B, #0
ld   A, #0
ld   IX, #LCDSEGDIGIT1
LCD_DISP_3DIG_SEG_DATA_WITH_ZERO_SUP
```

LCD_DISP_2DIG_SEG_DATA_SUP_ZERO_MSD

Description Display a 2-digit BCD data with zero suppression on the MSD.

Usage LCD_DISP_2DIG_SEG_DATA_SUP_ZERO_MSD

Assumptions None

Input

A	-	packed 10's and 1's digit data
IX	-	Leftmost 9-segment digit position where we wish to display the data

Output None

Destroys A, B, L, IX

Example

```
; Display the BCD value 0x00. After the macro call, this will be
; displayed as 1 space followed by a “0”
ld   A, #0
ld   IX, #LCDSEGDIGIT1
LCD_DISP_2DIG_SEG_DATA_SUP_ZERO_MSD
```

LCD_DISP_3DIG_SEG_DATA_NO_LSD_SUP

Description Display a 3-digit segmented BCD data with zero suppression on all digits except the LSD.

Usage LCD_DISP_3DIG_SEG_DATA_NO_LSD_SUP

Assumptions None

Input

B	-	100's digit BCD data
A	-	packed 10's and 1's digit data
IX	-	Leftmost 9-segment digit position where we wish to display the data.

Output None

Destroys A, B, L, IX

Example

```
; Display the BCD value 0x000. After the macro call, this will be
; displayed as 2 spaces followed by a “0”
ld   B, #0
ld   A, #0
ld   IX, #LCDSEGDIGIT1
LCD_DISP_3DIG_SEG_DATA_NO_LSD_SUP
```

LCD_DISP_2DIG_SEG_DATA_NO_ZERO_SUP

Description Display a 2-digit BCD data with no zero suppression on both digits.

Usage LCD_DISP_2DIG_SEG_DATA_NO_ZERO_SUP

Assumptions None

Input

A	-	packed 10's and 1's digit data
IX	-	Leftmost 9-segment digit position where we wish to display the data.

Output None

Destroys A, B, L, IX

Example

```
; Display the BCD value 0x00. After the macro call, this will be displayed
;as “00”
ld   A, #0
ld   IX, #LCDSEGDIGIT1
LCD_DISP_2DIG_SEG_DATA_NO_ZERO_SUP
```

LCD_DISP_3DIG_SEG_DATA_NO_ZERO_SUP

Description Display a 3-digit segmented BCD data with no zero suppression on all digits.

Usage LCD_DISP_3DIG_SEG_DATA_NO_ZERO_SUP

Assumptions None

Input

B	-	100's digit BCD data
A	-	packed 10's and 1's digit data
IX	-	Leftmost 9-segment digit position where we wish to display the data.

Output None

Destroys A, B, L, IX

Example

```
; Display the BCD value 0x000. After the macro call, this will be
; displayed as “000”
ld   B, #0
ld   A, #0
ld   IX, #LCDSEGDIGIT1
LCD_DISP_3DIG_SEG_DATA_NO_ZERO_SUP
```

LCD_DISP_4DIG_SEG_DATA_WITH_ZERO_SUP

Description Display a 4-digit segmented data with zero suppression in the leftmost 4 9-segment display

Usage LCD_DISP_4DIG_SEG_DATA_WITH_ZERO_SUP

Assumptions None

Input

BA	-	BCD 4 digit packed data to be displayed. B contains the 1000s and 100s digits, and A contains the 10s and 1s digits.

Output None

Destroys A, B, L, IX

Example

```
; Display the BCD value 0x0001. After the macro call, this will be
; displayed as 3 spaces followed by a “1” in digits 1 through 4
ld   B, #0
ld   A, #01H
LCD_DISP_4DIG_SEG_DATA_WITH_ZERO_SUP
```

LCD_DISP_SMALL_PROP_WIDTH_2DIG_DM_DATA_SUP_ZERO

Description Display small-font, proportional-width 2-digit data with zero suppression on both digits.

Usage LCD_DISP_SMALL_PROP_WIDTH_2DIG_DM_DATA_SUP_ZERO

Assumptions None

Input

A	-	Packed 10's and 1's digit BCD data
IX	-	Leftmost dot matrix column and line where we wish to display the data.

Output None

Destroys BA, HL, IX

Example

```
; Display the BCD value 0x01 in Line 2, column20. After the macro call,
; this will be displayed as 1 space followed by “1”
ld   A, #01H
ld   IX, #LCDMAINDMLINE2COL20
LCD_DISP_SMALL_PROP_WIDTH_2DIG_DM_DATA_SUP_ZERO
```

LCD_DISP_SMALL_FIXED_WIDTH_2DIG_DM_DATA_SUP_ZERO

Description Display small-font, fixed-width 2-digit data with zero suppression on both digits.

Usage LCD_DISP_SMALL_FIXED_WIDTH_2DIG_DM_DATA_SUP_ZERO

Assumptions None

Input

A	-	Packed 10's and 1's digit BCD data
IX	-	Leftmost dot matrix column and line where wish to display the data.

Output None

Destroys BA, HL, IX

Example

```
; Display the BCD value 0x00 in Line 1, column23. After the macro call,
; this will be displayed as 2 spaces
ld   A, #00H
ld   IX, #LCDMAINDMLINE1COL23
LCD_DISP_SMALL_FIXED_WIDTH_2DIG_DM_DATA_SUP_ZERO
```

LCD_DISP_SMALL_PROP_WIDTH_3DIG_DM_DATA_SUP_ZERO

Description Display small-font, proportional-width 3-digit data with zero suppression on all leading digits.

Usage LCD_DISP_SMALL_PROP_WIDTH_3DIG_DM_DATA_SUP_ZERO

Assumptions None

Input

B	-	100's digit BCD data
A	-	Packed 10's and 1's digit BCD data
IX	-	leftmost dot matrix column and line where we wish to display the data.

Output None

Destroys BA, HL, IX

Example

```
; Display the BCD value 0x007 in Line 1, column23. After the macro call,
; this will be displayed as 2 spaces followed by a “7”.
ld   A, #07H
ld   B, #00H
ld   IX, #LCDMAINDMLINE1COL23
LCD_DISP_SMALL_PROP_WIDTH_3DIG_DM_DATA_SUP_ZERO
```

LCD_DISP_SMALL_FIXED_WIDTH_3DIG_DM_DATA_SUP_ZERO

Description Display small-font, fixed-width 3-digit data with zero suppression on all leading digits.

Usage LCD_DISP_SMALL_FIXED_WIDTH_3DIG_DM_DATA_SUP_ZERO

Assumptions None

Input

B	-	100's digit BCD data
A	-	Packed 10's and 1's digit BCD data
IX	-	Leftmost dot matrix column and line where we wish to display the data.

Output None

Destroys BA, HL, IX

Example

```
; Display the BCD value 0x007 in Line 1, column23. After the macro call,
; this will be displayed as 2 spaces followed by a “7”.
ld   A, #07H
ld   B, #00H
ld   IX, #LCDMAINDMLINE1COL23
LCD_DISP_SMALL_FIXED_WIDTH_3DIG_DM_DATA_SUP_ZERO
```

LCD_DISP_SMALL_PROP_WIDTH_2DIG_DM_DATA_SUP_ZERO_MSD

Description Display small-font, proportional-width 2-digit data with zero suppression on the MSD.

Usage LCD_DISP_SMALL_PROP_WIDTH_2DIG_DM_DATA_SUP_ZERO_MSD

Assumptions None

Input

A	-	Packed 10's and 1's digit BCD data
IX	-	Leftmost dot matrix column and line where we wish to display the data..

Output None

Destroys BA, HL, IX

Example

```
; Display the BCD value 0x00 in Line 1, column23. After the macro call,\
; this will be displayed as 1 space followed by “0”
ld   A, #00H
ld   IX, #LCDMAINDMLINE1COL23
LCD_DISP_SMALL_PROP_WIDTH_2DIG_DM_DATA_SUP_ZERO_MSD
```

LCD_DISP_SMALL_FIXED_WIDTH_2DIG_DM_DATA_SUP_ZERO_MSD

Description Display small-font, fixed-width 2-digit data with zero suppression on the MSD.

Usage LCD_DISP_SMALL_FIXED_WIDTH_2DIG_DM_DATA_SUP_ZERO_MSD

Assumptions None

Input

A	-	Packed 10's and 1's digit BCD data
IX	-	Leftmost dot matrix column and line where we wish to display the data

Output None

Destroys BA, HL, IX

Example

```
; Display the BCD value 0x00 in Line 1, column23. After the macro call,
; this will be displayed as 1 space followed by “0”
ld   A, #00H
ld   IX, #LCDMAINDMLINE1COL23
LCD_DISP_SMALL_FIXED_WIDTH_2DIG_DM_DATA_SUP_ZERO_MSD
```

LCD_DISP_SMALL_PROP_WIDTH_3DIG_DM_DATA_NO_LSD_SUP

Description Display small-font, proportional-width 3-digit data with suppression on leading zeros except the LSD.

Usage LCD_DISP_SMALL_PROP_WIDTH_3DIG_DM_DATA_NO_LSD_SUP

Assumptions None

Input

B	-	100's digit BCD data
A	-	Packed 10's and 1's digit BCD data
IX	-	Leftmost dot matrix column and line where we wish to display the data

Output None

Destroys BA, HL, IX

Example

```
; Display the BCD value 0x000 in Line 1, column23. After the macro call,
this ; will be displayed as 2 spaces followed by a “0”.
ld   A, #00H
ld   B, #00H
ld   IX, #LCDMAINDMLINE1COL23
LCD_DISP_SMALL_PROP_WIDTH_3DIG_DM_DATA_NO_LSD_SUP
```

LCD_DISP_SMALL_FIXED_WIDTH_3DIG_DM_DATA_NO_LSD_SUP

Description Display small-font, fixed-width 3-digit data with suppression on leading zeros except LSD.

Usage LCD_DISP_SMALL_FIXED_WIDTH_3DIG_DM_DATA_NO_LSD_SUP

Assumptions None

Input

B	-	100's digit BCD data
A	-	Packed 10's and 1's digit BCD data
IX	-	Leftmost dot matrix column and line where we wish to display the data

Output None

Destroys BA, HL, IX

Example

```
; Display the BCD value 0x000 in Line 1, column23. After the macro call,
; this will be displayed as 2 spaces followed by a “0”.
ld   A, #00H
ld   B, #00H
ld   IX, #LCDMAINDMLINE1COL23
LCD_DISP_SMALL_FIXED_WIDTH_3DIG_DM_DATA_NO_LSD_SUP
```

LCD_DISP_SMALL_PROP_WIDTH_2DIG_DM_DATA_NO_ZERO_SUP

Description Display small-font, fixed-width 2-digit data with no suppression on both zeros.

Usage LCD_DISP_SMALL_PROP_WIDTH_2DIG_DM_DATA_NO_ZERO_SUP

Assumptions None

Input

A	-	Packed 10's and 1's digit BCD data
IX	-	Leftmost dot matrix column and line where we wish to display the data

Output None

Destroys BA, HL, IX

Example

```
; Display the BCD value 0x00 in Line 1, column23. After the macro call,
; this will be displayed as “00”.
ld   A, #00H
ld   IX, #LCDMAINDMLINE1COL23
LCD_DISP_SMALL_PROP_WIDTH_2DIG_DM_DATA_NO_ZERO_SUP
```

LCD_DISP_SMALL_FIXED_WIDTH_2DIG_DM_DATA_NO_ZERO_SUP

Description Display small-font, fixed-width 2-digit data with no suppression on both zeros.

Usage LCD_DISP_SMALL_FIXED_WIDTH_2DIG_DM_DATA_NO_ZERO_SUP

Assumptions None

Input

A	-	Packed 10's and 1's digit BCD data
IX	-	Leftmost dot matrix column and line where we wish to display the data

Output None

Destroys BA, HL, IX

Example

```
; Display the BCD value 0x00 in Line 1, column23. After the macro call,
; this will be displayed as “00”.
ld   A, #00H
ld   IX, #LCDMAINDMLINE1COL23
LCD_DISP_SMALL_FIXED_WIDTH_2DIG_DM_DATA_NO_ZERO_SUP
```

LCD_DISP_SMALL_FIXED_WIDTH_3DIG_DM_DATA_NO_ZERO_SUP

Description Display small-font, fixed-width 3-digit data with no suppression on all zeros.

Usage LCD_DISP_SMALL_FIXED_WIDTH_3DIG_DM_DATA_NO_ZERO_SUP

Assumptions None

Input

B	-	100's digit BCD data
A	-	Packed 10's and 1's digit BCD data
IX	-	Leftmost dot matrix column and line where we wish to display the data

Output None

Destroys BA, HL, IX

Example

```
; Display the BCD value 0x000 in Line 1, column23. After the macro call,
this ; will be displayed as “000”.
ld   B, #00H
ld   A, #00H
ld   IX, #LCDMAINDMLINE1COL23
LCD_DISP_SMALL_FIXED_WIDTH_3DIG_DM_DATA_NO_ZERO_SUP
```

Large-Font Numeric Display

LCD_DISP_BIG_2DIGIT_DM_DATA_SUP_ZERO

Description Display a large-font, 2-digit DM data with zero suppression on leading digit positions.

Usage LCD_DISP_BIG_2DIGIT_DM_DATA_SUP_ZERO

Assumptions None

Input

A	-	Packed 10's and 1's digit BCD data
IX	-	Leftmost dot matrix column where we wish to display the data.

Output None

Destroys BA, HL, IX

Example

```
; Display the BCD value 0x08 in column23. After the macro call, this will
; be displayed as “_8” (space followed by 8)
ld   A, #08H
ld   IX, #LCDBIGCHARDMCOL23
LCD_DISP_BIG_2DIGIT_DM_DATA_SUP_ZERO
```

LCD_DISP_BIG_3DIGIT_DM_DATA_SUP_ZERO

Description Display a large-font, 3-digit DM data with zero suppression on leading digit positions.

Usage LCD_DISP_BIG_3DIGIT_DM_DATA_SUP_ZERO

Assumptions None

Input

B	-	100's digit BCD data
A	-	Packed 10's and 1's digit BCD data
IX	-	Leftmost dot matrix column where we wish to display the data.

Output None

Destroys BA, HL, IX

Example

```
; Display the BCD value 0x000 in Line 1, column23
; After the macro call, this will be displayed as 3 spaces.
ld   B, #00H
ld   A, #00H
ld   IX, #LCDBIGCHARDMCOL23
LCD_DISP_BIG_3DIGIT_DM_DATA_SUP_ZERO
```

LCD_DISP_BIG_2DIGIT_DM_DATA_SUP_ZERO_MSD

Description Display a large-font, 2-digit DM data with zero suppression on the MSD.

Usage LCD_DISP_BIG_2DIGIT_DM_DATA_SUP_ZERO_MSD

Assumptions None

Input

A	-	Packed 10's and 1's digit BCD data
IX	-	Leftmost dot matrix column where we wish to display the data.

Output None

Destroys BA, HL, IX

Example

```
; Display the BCD value 0x00 in column23. After the macro call, this will
; be displayed as 1 space followed by a “0”.
ld   A, #00H
ld   IX, #LCDBIGCHARDMCOL23
LCD_DISP_BIG_2DIGIT_DM_DATA_SUP_ZERO_MSD
```

LCD_DISP_BIG_3DIGIT_DM_DATA_NO_LSD_SUP

Description Display a large-font, 3-digit DM data with zero suppression on all digits except the LSD.

Usage LCD_DISP_BIG_3DIGIT_DM_DATA_NO_LSD_SUP

Assumptions None

Input

B	-	100's digit BCD data
A	-	Packed 10's and 1's digit BCD data
IX	-	Leftmost dot matrix column where we wish to display the data.

Output None

Destroys BA, HL, IX

Example

```
; Display the BCD value 0x000 in column23. After the macro call, this will
; be displayed as 2 spaces followed by “0”.
ld   B, #00H
ld   A, #00H
ld   IX, #LCDBIGCHARDMCOL23
LCD_DISP_BIG_3DIGIT_DM_DATA_NO_LSD_SUP
```

LCD_DISP_BIG_2DIGIT_DM_DATA_NO_ZERO_SUP

Description Display a large-font, 2-digit DM data with no zero suppression on all digits.

Usage LCD_DISP_BIG_2DIGIT_DM_DATA_NO_ZERO_SUP

Assumptions None

Input

A	-	Packed 10's and 1's digit BCD data
IX	-	Leftmost dot matrix column where we wish to display the data.

Output None

Destroys BA, HL, IX

Example

```
; Display the BCD value 0x00 in column23. After the macro call, this will
; be displayed as “00”.
ld   A, #00H
ld   IX, #LCDBIGCHARDMCOL23
LCD_DISP_BIG_2DIGIT_DM_DATA_NO_ZERO_SUP
```

LCD_DISP_BIG_3DIGIT_DM_DATA_NO_ZERO_SUP

Description Display a large-font, 3-digit DM data with no zero suppression on all digits.

Usage LCD_DISP_BIG_3DIGIT_DM_DATA_NO_ZERO_SUP

Assumptions None

Input

B	-	100's digit BCD data
A	-	Packed 10's and 1's digit BCD data
IX	-	Leftmost dot matrix column where we wish to display the data.

Output None

Destroys BA, HL, IX

Example

```
; Display the BCD value 0x000 in column23. After the macro call, this will
; be displayed as “000”.
ld   B, #00H
```

```
ld   A, #00H
ld   IX, #LCDBIGCHARDMCOL23
LCD_DISP_BIG_3DIGIT_DM_DATA_NO_ZERO_SUP
```

Message Display

LCD_DISP_SEG_LINE_MSG

Description Display a 6-character segmented message.

Usage LCD_DISP_SEG_LINE_MSG

Assumptions None

Input IY - starting address of the message

The message structure is formatted as follows:

```
db  char_1, char_2, … , char_6
```

For example:

```
Seg_Message:
      db SEG_S, SEG_P, SEG_L, SEG_I, SEG_T, SEG_SPACE_
```

Here, IY would be set to whatever address the label "`Seg_Message`" assembled to.

Output None

Destroys A, B, L, IX

Example

```
; Display "SPLIT " message
ld   IY, #lcdSegMsg_SPLIT
LCD_DISP_SEG_LINE_MSG
```

LCD_DISP_FORMATTED_SMALL_PROP_WIDTH_DM_MSG

Description Display a formatted, small-font, proportional-width dot-matrix message.

Usage LCD_DISP_FORMATTED_SMALL_PROP_WIDTH_DM_MSG

Assumptions None

Input IY - starting address of the message

The message structure is formatted as follows:

```
dw  {display_address ,ie columnn and line to start message at }
db  {number_of_characters}, char_1, char_2, … , char_N
```

For example:

```
dw  LCDMAINDMLINE2COL4
db  5, DM5_S, DM5_P, DM5_L, DM5_I, DM5_T
```

Output None

Destroys BA, HL, IX

Example

```
; Display the stored message “ENTER” in line 1, column 8
ld   IY, #lcdSmDMMsg_ENTER
LCD_DISP_FORMATTED_SMALL_PROP_WIDTH_DM_MSG
```

LCD_DISP_FORMATTED_SMALL_FIXED_WIDTH_DM_MSG

Description Display a formatted, small-font, fixed-width dot-matrix message.

Usage LCD_DISP_FORMATTED_SMALL_FIXED_WIDTH_DM_MSG

Assumptions None

Input IY - starting address of the message

The message structure is formatted as follows:

```
dw  {display_address ie, columnn and line to start message at }
db  {number_of_characters}, char_1, char_2, … , char_N
```

For example:

```
DM_Message:
      dw  LCDMAINDMLINE2COL4
      db  5, DM5_S, DM5_P, DM5_L, DM5_I, DM5_T
```

Here, IY would be set to whatever address the label “`DM_Message`” assembled to.

Output None

Destroys BA, HL, IX

Example

```
; Display the stored message “ENTER” in line 1, column 8
ld   IY, #lcdSmDMMsg_ENTER
LCD_DISP_FORMATTED_SMALL_FIXED_WIDTH_DM_MSG
```

LCD_DISP_UNFORMATTED_SMALL_PROP_WIDTH_DM_MSG

Description Display an unformatted, small-font, proportional-width dot-matrix message.

Usage LCD_DISP_UNFORMATTED_SMALL_PROP_WIDTH_DM_MSG

Assumptions None

Input

IY	-	starting address of the message
IX	-	starting display address
B	-	Number of characters to be displayed

The message structure is formatted as follows:

```
db  char_1, char_2, … , char_N
```

For example:

DM_Message:

```
db  DM5_S, DM5_P, DM5_L, DM5_I, DM5_T
```

Here, IY would be set to whatever address the label "DM_Message" assembled to.

Output None

Destroys BA, HL, IX

Example

```
; Display the stored message "ENTER" in line 1, column 8
ld   IY, #(lcdSmDMMsg_ENTER +3)
ld   IX, #LCDMAINDMLINE1COL8
ld   B, #5
LCD_DISP_UNFORMATTED_SMALL_PROP_WIDTH_DM_MSG
```

LCD_DISP_UNFORMATTED_SMALL_FIXED_WIDTH_DM_MSG

Description Display a unformatted, small-font, fixed-width dot-matrix message.

Usage LCD_DISP_UNFORMATTED_SMALL_FIXED_WIDTH_DM_MSG

Assumptions None

Input

IY	-	starting address of the message
IX	-	Leftmost address of the LCD, column and line, where we wish to display the message
B	-	Number of characters to be displayed

The message structure is formatted as follows:

```
db  char_1, char_2, … , char_N
```

For example:
DM_Message:

```
db  DM5_S, DM5_P, DM5_L, DM5_I, DM5_T
```

Here, IY would be set to whatever address the label "DM_Message" assembled to.

Output None

Destroys BA, HL, IX

Example

```
; Display the stored message "ENTER" in line 1, column 8
ld   IY, #(lcdSmDMMsg_ENTER +3)
ld   IX, #LCDMAINDMLINE1COL8
ld   B, #5
LCD_DISP_UNFORMATTED_SMALL_FIXED_WIDTH_DM_MSG
```

Banner Message Display

LCD_DISP_BANNER_MSG

Description Display a banner message.

Usage LCD_DISP_BANNER_MSG

Assumptions None

Input IY - starting address of the banner message structure

The banner message has control codes embedded in the message that either indicates the column position and the end of the message. The table below shows the control codes:

Control Code	Description
LCDBANNER_COLn	Signals the starting column to display the message where ***n*** indicates the column position. This is always the first byte in the message. If the control code is present in the middle of the message array, this will signify the starting column in the second line.
LCD_END_BANNER	Signals the end of the banner message.

Sample code for a two-line mode banner message:

```
lcdBannerMsg_TIME_OF_DAY:
       db   LCDBANNER_COL12
       db   DM5_T, DM5_I, DM5_M, DM5_E
       db   LCDBANNER_COL6
```

```
        db   DM5_O, DM5_F, DM5_SPACE, DM5_D, DM5_A, DM5_Y
        db   LCD_END_BANNER

        IY would be set to the address lcdBannerMsg_TIME_OF_DAY assembles to.
```

Sample code for a one-line mode banner message:

```
lcdBannerMsg_CHRONO:
        db   LCDBANNER_COL5
        db   DM5_C, DM5_H, DM5_R, DM5_O, DM5_N, DM5_O
        db   LCD_END_BANNER

        IY would be set to the address lcdBannerMsg_CHRONO assembles to.
```

Sample code for a one-line mode banner message in the second line:

```
lcdBannerMsg_CHRONO:
        db   LCDBANNER_COL1
        db   LCDBANNER_COL5
        db   DM5_C, DM5_H, DM5_R, DM5_O, DM5_N, DM5_O
        db   LCD_END_BANNER

        IY would be set to the address lcdBannerMsg_CHRONO assembles to.
```

Output None

Destroys BA, HL, IX

Example See explanations above under “INPUT”.

Large-Font Message Display

LCD_DISP_FORMATTED_BIG_FONT_DM_MSG

Description Display a formatted, large-font, dot matrix message.

Usage LCD_DISP_FORMATTED_BIG_FONT_DM_MSG

Assumptions None

Input IY - starting address of the message

The message structure is formatted as follows:

```
dw  {display_address, ie, leftmost column message starts at }
db  {number_of_characters}, char_1, char_2, … , char_N
```

For example:

DM_BIG_Message:

```
dw  LCDBIGCHARDMCOL3
db  5, DM8_S, DM8_P, DM8_L, DM8_I, DM8_T

IY would be set to the address DM_BIG_Message assembles to.
```

Output None

Destroys BA, HL, IX

Example

```
; Display the stored message “SPLIT” in column 1
ld   IY, #lcdBigDMMsg_SPLIT
LCD_DISP_FORMATTED_BIG_FONT_DM_MSG
```

LCD_DISP_UNFORMATTED_BIG_FONT_DM_MSG

Description Display an unformatted, large-font, dot matrix message.

Usage LCD_DISP_UNFORMATTED_BIG_FONT_DM_MSG

Assumptions None

Input

IY	-	starting address of the message
IX	-	Leftmost column on LCD that the message is displayed to.
B	-	Number of characters to be displayed

The message structure is formatted as follows:

```
db  char_1, char_2, … , char_N
```

For example:

DM_BIG_Message:

```
db  DM8_S, DM8_L, DM8_I, DM8_T

IY would be set to the address DM_BIG_Message assembles to.
```

Output None

Destroys BA, HL, IX

Example

```
; Display the stored message “SPLIT” in column 1
ld   IY, #( lcdBigDMMsg_SPLIT +3)
ld   IX, #LCDBIGCHARDMCOL1
ld   B, #5
LCD_DISP_UNFORMATTED_BIG_FONT_DM_MSG
```

Icon/Flag Blinking

LCD_SET_BLINK_FLAG_STATUS

Description Sets a blink flag bit when an application requests flag to be blinked in foreground at a 2Hz rate.

Usage LCD_SET_BLINK_FLAG_STATUS {*flag_name*}

Assumptions None

Input *flag_name* - Resource icon to be blinked

The following definitions are available for the *flag_name*:

Constant
CHR_FLAG
MOON_FLAG
ACLK_FLAG
ARROW_FLAG
TAIL_FLAG
TMR_FLAG
NOTE_FLAG
P_FLAG
A_FLAG
L_FLAG

Output None

Destroys BA

Example

```
; To blink only one icon:
; Blink the CHRONO flag and then Set the core request flags to
; enable 2Hz blinking
LCD_SET_BLINK_FLAG_STATUS   CHR_FLAG
CORE_ENABLE_2HZ_BLINKING


; Alternative API. . .
; Use the API below that replaces the two lines in the example above
CORE_REQ_BLINK_2HZ CHR_FLAG



; To blink more than one icon:
LCD_SET_BLINK_FLAG_STATUS   CHR_FLAG
LCD_SET_BLINK_FLAG_STATUS   TMR_FLAG
CORE_ENABLE_2HZ_BLINKING
```

LCD_CLR_BLINK_FLAG_STATUS

Description Clears a blink flag bit to stop blinking the specified flag.

Usage LCD_CLR_BLINK_FLAG_STATUS {*flag_name*}

Assumptions None

Input *flag_name* - flag whose blinking is to be halted

The following definitions are available for the *flag_name*:

Constant
CHR_FLAG
MOON_FLAG
ACLK_FLAG
ARROW_FLAG
TAIL_FLAG
TMR_FLAG
NOTE_FLAG
P_FLAG
A_FLAG
L_FLAG

Output None

Destroys BA

Example

```
; Stop blinking the CHRONO and TIMER flags
LCD_CLR_BLINK_FLAG_STATUS   CHR_FLAG
LCD_CLR_BLINK_FLAG_STATUS   TMR_FLAG
```

LCD_CANCEL_ALL_BLINK_FLAGS

Description Clears all the blink status flags.

Usage LCD_CANCEL_ALL_BLINK_FLAGS

Assumptions None

Input None

Output None

Destroys BA

Example

```
; Stop all flag blinking
LCD_CANCEL_ALL_BLINK_FLAGS
```

Blinking

LCD_WRITE_4HZ_GEN_BLINK_DISP_ROUTINE_ADDR

Description Specify the display routine for the 4Hz generic blinking.

Usage LCD_WRITE_4HZ_GEN_BLINK_DISP_ROUTINE_ADDR {*display_routine*}

Assumptions None

Input *display_routine* - address of the function that displays the required data

For example, if we wish to blink the ALARM flag at 4Hz, we could set up two routines, one to turn the flag ON, and one to turn the flag OFF, and call them alternately at a 4Hz rate.

Below is the routine to turn it on;

```
BlinkAlarm_ON:
  LCD_UPD_ACLK_FLAG ON
  ret
```

Similarly, to turn the flag OFF;

Below is the routine to turn it on;

```
BlinkAlarm_Off:
  LCD_UPD_ACLK_FLAG OFF
  ret
```

To execute these routines, we need to load their addresses into the buffer used to tell the generic 4Hz blink routine what routine should be called to do the blinking. We call the macro described here to load the ON routine, with the address the routine assembles at as the parameter.

```
LCD_WRITE_4HZ_GEN_BLINK_DISP_ROUTINE_ADDR BlinkAlarm_ON
LCD_WRITE_4HZ_GEN_BLINK_CLR_ROUTINE_ADDR BlinkAlarm_OFF
```

Output None

Destroys BA

Example

```
; Load address of alarm routine into BLINK ON area.
LCD_WRITE_4HZ_GEN_BLINK_DISP_ROUTINE_ADDR BlinkAlarm_ON
```

LCD_WRITE_4HZ_GEN_BLINK_CLR_ROUTINE_ADDR

Description Specify the clear routine for the 4Hz generic blinking.

Usage LCD_WRITE_4HZ_GEN_BLINK_CLR_ROUTINE_ADDR {*clear_routine*}

Assumptions None

Input *clear _routine* - address of the function that clears the required data

For example, if we wish to blink the ALARM flag at 4Hz, we could set up two routines, one to turn the flag ON, and one to turn the flag OFF, and call them alternately at a 4Hz rate.

Below is the routine to turn it on;

```
BlinkAlarm_ON:
  LCD_UPD_ACLK_FLAG ON
  ret
```

Similarly, to turn the flag OFF;

Below is the routine to turn it on;

```
BlinkAlarm_Off:
  LCD_UPD_ACLK_FLAG OFF
  ret
```

To execute these routines, we need to load their addresses into the buffer used to tell the generic 4Hz blink routine what routine should be called to do the blinking. We call the macro described here to load the ON routine, with the address the routine assembles at as the parameter.

```
LCD_WRITE_4HZ_GEN_BLINK_DISP_ROUTINE_ADDR BlinkAlarm_ON
LCD_WRITE_4HZ_GEN_BLINK_CLR_ROUTINE_ADDR BlinkAlarm_OFF
```

Output None

Destroys BA

Example

```
; Load the CLEAR routine address for the ALARM flag
LCD_WRITE_4HZ_GEN_BLINK_CLR_ROUTINE_ADDR BlinkAlarm_OFF
```

LCD_WRITE_4HZ_GEN_BLINK_POSITION

Description Specify the position for blinking.

Usage LCD_WRITE_4HZ_GEN_BLINK_POSITION {*blink_position*}

Assumptions None

Input *blink_position* - Position for blinking data/

This routine loads an address on the LCD into a buffer in RAM. This location can be used by the

generic 4Hz blink routine to blink a specific address or group of addresses.
The blink routines the user writes are NOT restricted to using this address only. The routine may specify any address within itself.

Output None

Destroys BA

Example
```
; Tell the 4Hz routine that it will blink the ALARM flag.
LCD_WRITE_4HZ_GEN_BLINK_POSITION      ACLK_ADDR
```

LCD_WRITE_4HZ_GEN_BLINK_DISP_ROUTINE_PRELOADED

Description Specifies in register BA the display routine used for the generic blinking.

Usage LCD_WRITE_4HZ_GEN_BLINK_DISP_ROUTINE_PRELOADED

Assumptions None

Input BA - 16-bit address of the display routine

Output None

Destroys BA

Example
```
; Set BLINK ON routine address into BA
ld    BA, # BlinkAlarm_ON
; Call MACRO to load BA into address area
LCD_WRITE_4HZ_GEN_BLINK_DISP_ROUTINE_PRELOADED
```

LCD_WRITE_4HZ_GEN_BLINK_CLR_ROUTINE_PRELOADED

Description Specifies in register BA the clear routine used for the generic blinking.

Usage LCD_WRITE_4HZ_GEN_BLINK_CLR_ROUTINE_PRELOADED

Assumptions None

Input BA - 16-bit address of the clear routine

Output None

Destroys BA

Example
```
; Set BLINK OFF routine address into BA
ld    BA, # BlinkAlarm_OFF
; Call MACRO to load BA into address area
```

```
LCD_WRITE_4HZ_GEN_BLINK_CLR_ROUTINE_PRELOADED
```

LCD_WRITE_4HZ_GEN_BLINK_POSITION_PRELOADED

Description Specifies in register BA the blink position on the LCD used for the generic blinking.

Usage LCD_WRITE_4HZ_GEN_BLINK_POSITION_PRELOADED

Assumptions None

Input BA - 16-bit blink positions

Output None

Destroys BA

Example

```
; Load BLINK address into BA
ld    BA, # ACLK_ADDR
; Call MACRO to load BA into position area
LCD_WRITE_4HZ_GEN_BLINK_POSITION_PRELOADED
```

LCD_DISP_4HZ_DATA_FIRST

Description Display data first when blinking generic data using the 4hz update.

Usage LCD_DISP_4HZ_DATA_FIRST

Assumptions None

Input None

Output None

Destroys HL

Example

```
; Setup 4Hz blinking so we display data on the first blink, NOT the cleared
; position.
LCD_DISP_4HZ_DATA_FIRST
```

Scrolling

The message to be scrolled should have the DM5_SENTINEL as the last byte. This will indicate the end of the scrolling message. The code section below shows an example of a message than can be scrolled.

```
; Message “FOUR SCORES AND SEVEN YEARS AGO” + SENTINEL

GettysburgMessage:

   db     DM5_F, DM5_O, DM5_U, DM5_R, DM5_SPACE
   db     DM5_S, DM5_C, DM5_O, DM5_R, DM5_E, DM5_S, DM5_SPACE
   db     DM5_A, DM5_N, DM5_D, DM5_SPACE
   db     DM5_S, DM5_E, DM5_V, DM5_E, DM5_N, DM5_SPACE
   db     DM5_Y, DM5_E, DM5_A, DM5_R, DM5_S, DM_SPACE
   db     DM5_A, DM5_G, DM5_O
   db     DM5_SENTINEL
```

LCD_SCROLL_RAM_OR_ROM_MSG_MAIN_DM_LINE1

Description Scroll a message from the ROM or RAM in the 1st line of the main DM area.

Usage LCD_SCROLL_RAM_OR_ROM_MSG_MAIN_DM_LINE1 {EVENT_ON|EVENT_OFF}

Assumptions None

Input

IY	-	starting address of the message to be scrolled

The macro parameter:

EVENT_ON	-	generate a 'COREEVENT_END_OF_SCROLLING_MESS' event when the entire message has been scrolled
EVENT_OFF	-	Doesn’t generate an event

Output None

Destroys BA, HL, IX

Example

```
; Scroll the message given in the example, FOUR SCORE AND SEVEN YEARS AGO.
; Get an event after the scrolling is completed.
ld   IY, #GettysburgMessage
LCD_SCROLL_RAM_OR_ROM_MSG_MAIN_DM_LINE1     EVENT_ON
```

LCD_SCROLL_RAM_OR_ROM_MSG_MAIN_DM_LINE2

Description Scroll a message from the ROM or RAM in the 2nd line of the main DM area.

Usage LCD_SCROLL_RAM_OR_ROM_MSG_MAIN_DM_LINE2 {EVENT_ON|EVENT_OFF}

Assumptions None

Input IY - starting address of the message to be scrolled

The macro parameter:

EVENT_ON	-	generate a 'COREEVENT_END_OF_SCROLLING_MESS' event when the entire message has been scrolled
EVENT_OFF	-	Doesn't generate an event

Output None

Destroys BA, HL, IX

Example

```
; Scroll the message given in the example, FOUR SCORE AND SEVEN YEARS AGO.
; Non't get an event after the scrolling is completed.
ld   IY, #GettysburgMessage
LCD_SCROLL_RAM_OR_ROM_MSG_MAIN_DM_LINE2     EVENT_OFF
```

LCD_PAUSE_SCROLLING

Description Pause scrolling.

Usage LCD_PAUSE_SCROLLING

Assumptions None

Input None

Output None

Destroys HL

Example

```
; Temporarily halt the scrolling operation
LCD_PAUSE_SCROLLING
```

LCD_RESUME_SCROLLING

Description Resume scrolling operation.

Usage LCD_RESUME_SCROLLING

Assumptions None

Input None

Output None

Destroys HL, B

Example
```
; Restart the scroliing operation
LCD_RESUME_SCROLLING
```

LCD_SCROLL_MSG_LEFT

Description Scroll a message to the left.

Usage LCD_SCROLL_MSG_LEFT

Assumptions Automatic scrolling is not active and all scroll parameters are set.

Input A - number of pixel columns to be scrolled { 1 – 14 }

Output None

Destroys BA, HL, IX

Example
```
; Pause the message
LCD_PAUSE_SCROLLING
; Now scroll the message left by 5 pixel columns
ld   A, #5
LCD_SCROLL_MSG_LEFT
```

LCD_SCROLL_MSG_RIGHT

Description Scroll a message to the right.

Usage LCD_SCROLL_MSG_RIGHT

Assumptions Automatic scrolling is not active and all scroll parameters are set.

Input A - number of pixel columns to be scrolled { 1 – 14 }

Output None

Destroys BA, HL, IX

Example
```
; Pause the message
LCD_PAUSE_SCROLLING
; Now scroll the message right by 5 pixel columns
ld   A, #5
LCD_SCROLL_MSG_RIGHT
```

Primary Mode Icon Resource

LCD_UPDATE_TOD_FLAG_RESOURCE_STATE

Description Update the state of a TOD flag resource.

Usage LCD_UPDATE_TOD_FLAG_RESOURCE_STATE {*flag_name*}, {*flag_state*}

Assumptions None

Input

A	-	The application index of calling function. This is used to check if the specified application index is the owner of the lcd icon resource.
flag_name	-	Resource icon to be updated
flag_state	-	Specifies how the icon is to be displayed.

The following definitions are available for the *flag_name*:

Constant
MOON_RSRC_FLAG
BATT_RSRC_FLAG
NOTE_RSRC_FLAG
HAND_RSRC_FLAG
ACLK_RSRC_FLAG
ARROW_RSRC_FLAG
TAIL_RSRC_FLAG
TMR_RSRC_FLAG
CHR_RSRC_FLAG
P_RSRC_FLAG
A_RSRC_FLAG
L_RSRC_FLAG

The following definitions are available for the *flag_state*:

Constant
BLINK_ON
BLINKOFF_FLAGOFF
BLINKOFF_FLAGON
FLAG_ON
FLAG_OFF

Output None

Destroys BA

Example

```
; Update the status of the MOON flag to BLINKING
; Get the current APP Index.
; We assume that the application wishing to modify the status is the
; foreground application.
ld    A, [CORECurrentMode]
; Update the flag status
LCD_UPDATE_TOD_FLAG_RESOURCE_STATE    MOON_RSRC_FLAG,   BLINK_ON
```

LCD_UPDATE_TIMELINE_RESOURCE

Description Update the Timeline resource data owned by the application.

Usage LCD_UPDATE_TIMELINE_RESOURCE

Assumptions None

Input

A	-	Application Index of the application which wants to modify the resource
IY	-	Address of data to be copied into the resource status buffer. This data consists of one byte

Output None

Destroys BA, IX

Example

```
; Get the current APP Index.
; We assume that the application wishing to modify the status is the
; foreground application.
ld    A, [CORECurrentMode]
; Get the application's TIMELINE data
ld   IY, #APPTTimelineData
; Call the macro to alter the timeline status
LCD_UPDATE_TIMELINE_RESOURCE
```

Pixel Operations

LCD_DISPLAY_MAIN_DM_PIXEL

Description Display a pixel in the main DM area.

Usage LCD_DISPLAY_MAIN_DM_PIXEL

Assumptions None

Input

A	-	x-coordinate in the main DM area { 1 – 42 }
B	-	y-coordinate in the main DM area { 1 – 11 }

Output None

Destroys BA, HL

Example

```
; Turn on Pixel (x,y) = (22, 8)
ld   A, #22
ld   B, #8
LCD_DISPLAY_MAIN_DM_PIXEL
```

LCD_CLEAR_MAIN_DM_PIXEL

Description Clears a pixel in the main DM area.

Usage LCD_CLEAR_MAIN_DM_PIXEL

Assumptions None

Input
A - x-coordinate in the main DM area { 1 – 42 }
B - y-coordinate in the main DM area { 1 – 11 }

Output None

Destroys BA, HL

Example
```
; Turn off Pixel (x,y) = (22, 8)
ld   A, #22
ld   B, #8
LCD_CLEAR_MAIN_DM_PIXEL
```

LCD_GET_STATE_OF_PIXEL

Description Returns the status of a pixel column (displayed or cleared).

Usage LCD_GET_STATE_OF_PIXEL

Assumptions None

Input
A - x-coordinate in the main DM area { 1 – 42 }
B - y-coordinate in the main DM area { 1 – 11 }

Output
Zero Flag = 1 - if pixel is cleared
Zero Flag = 0 - if pixel is displayed

Destroys BA, HL

Example
```
; Get status of pixel
ld   A, #22
ld   B, #8
LCD_GET_STATE_OF_PIXEL
; Test Z flag to see if pixel is on
jr   Z, Exit
; If we get here, the pixel is OFF
```

Display Canned Messages

LCD_DISP_UPPER_DM_MSG_ONE_HALF

Description Display '1/2' on the upper DM area.

Usage LCD_DISP_UPPER_DM_MSG_ONE_HALF

Assumptions None

Input None

Output None

Destroys BA, HL

Example
```
;display “1/2”
LCD_DISP_UPPER_DM_MSG_ONE_HALF
```

LCD_DISP_UPPER_DM_MSG_TO

Description Display 'TO' on the upper DM area.

Usage LCD_DISP_UPPER_DM_MSG_TO

Assumptions None

Input None

Output None

Destroys BA, HL

Example
```
No Example
```

LCD_DISP_UPPER_DM_MSG_LO

Description Display 'LO' on the upper DM area.

Usage LCD_DISP_UPPER_DM_MSG_LO

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_HOLD_TO_SWITCH

Description Display 'HOLD TO SWITCH' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_HOLD_TO_SWITCH

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MSG_NO_APPT_UPCOMING

Description Display 'NO APPT UPCOMING' on the LCD.

Usage LCD_DISP_MSG_NO_APPT_UPCOMING

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_NO_OCCASION_UPCOMING

Description Display 'NO OCCASION UPCOMING' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_NO_OCCASION_UPCOMING

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_SET_TIME_SLASH_DATE

Description Display 'SET TIME/DATE' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_SET_TIME_SLASH_DATE

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_SHOW_DAY_OF_WEEK

Description Display 'SHOW DAY OF WEEK' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_SHOW_DAY_OF_WEEK

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_SHOW_WEEK_NUMBER

Description Display 'SHOW WEEK NUMBER' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_SHOW_WEEK_NUMBER

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_FREE_MEMORY_LAPS

Description Display 'FREE MEMORY LAPS' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_FREE_MEMORY_LAPS

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_TO_STORE_TURN_CROWN

Description Display 'TO STORE TURN CROWN' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_TO_STORE_TURN_CROWN

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_MEMORY_FULL

Description Display 'MEMORY FULL' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_MEMORY_FULL

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_HOLD_TO_CLR_ALL

Description Display 'HOLD TO CLR ALL' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_HOLD_TO_CLR_ALL

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_HOLD_TO_RESET

Description Display 'HOLD TO RESET' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_HOLD_TO_RESET

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_WORKOUT_STORED

Description Display 'WORKOUT STORED' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_WORKOUT_STORED

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_RECALL_SLASH_FORMAT

Description Display 'RECALL/FORMAT' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_RECALL_SLASH_FORMAT

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_FORMAT

Description Display 'FORMAT' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_FORMAT

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_HOLD_TO_CLEAR_WORKOUT

Description Display 'HOLD TO CLEAR WORKOUT' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_HOLD_TO_CLEAR_WORKOUT

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_HOLD_TO_CLEAR

Description Display 'HOLD TO CLEAR' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_HOLD_TO_CLEAR

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_YOU_ROCK

Description Display 'YOU ROCK' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_YOU_ROCK

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MSG_LAP_SPLIT

Description Display 'LAP' on the 9-segment area and 'SPLIT' on the main dot-matrix area.

Usage LCD_DISP_MSG_LAP_SPLIT

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MSG_SPLIT_LAP

Description Display 'SPLIT' on the 9-segment area and 'LAP' on the main dot-matrix area.

Usage LCD_DISP_MSG_SPLIT_LAP

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MSG_TIME_SPLIT

Description Display 'TIME' on the 9-segment area and 'SPLIT' on the main dot-matrix area.

Usage LCD_DISP_MSG_TIME_SPLIT

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_LARGE_DM_STOP

Description Display 'STOP' in large characters on the main DM area.

Usage LCD_DISP_LARGE_DM_STOP

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_LARGE_DM_TOTAL

Description Display 'TOTAL' in large characters on the main DM area.

Usage LCD_DISP_LARGE_DM_TOTAL

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MSG_TIME_LAP

Description Display 'TIME' on the 9-segment area and 'LAP' on the main dot-matrix area.

Usage LCD_DISP_MSG_TIME_LAP

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_SET_TIMER

Description Display 'SET TIMER' on the main dot-matrix area.

Usage LCD_DISP_SMALL_DM_MSG_SET_TIMER

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_UNUSED_ENTRY

Description Display 'UNUSED ENTRY' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_UNUSED_ENTRY

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_UNUSED_ENTRIES

Description Display 'UNUSED ENTRIES' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_UNUSED_ENTRIES

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_STOP_AT_END

Description Display 'STOP AT END' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_STOP_AT_END

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_REPEAT_AT_END

Description Display 'REPEAT AT END' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_REPEAT_AT_END

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_CHRONO_AT_END

Description Display 'CHRONO AT END' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_CHRONO_AT_END

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_SET_FORMAT

Description Display 'SET FORMAT' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_SET_FORMAT

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_SET_COUNTER

Description Display 'SET COUNTER' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_SET_COUNTER

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_COUNT_UP

Description Display 'COUNT UP' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_COUNT_UP

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_COUNT_DOWN

Description isplay 'COUNT DOWN' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_COUNT_DOWN

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_HOLD_TO_DELETE

Description Display 'HOLD TO DELETE' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_HOLD_TO_DELETE

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_SET_ALARM

Description Display 'SET ALARM' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_SET_ALARM

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_ENTRY_DELETED

Description Display 'ENTRY DELETED' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_ENTRY_DELETED

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_SET_APPT

Description Display 'SET APPT' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_SET_APPT

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_APPT_DATE

Description Display 'APPT DATE' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_APPT_DATE

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_1ST_APPT_DATE

Description Display '1ST APPT DATE' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_1ST_APPT_DATE

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_EDIT_NOTE

Description Display 'EDIT_NOTE' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_EDIT_NOTE

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_BUTTON_BEEP_ON

Description Display 'BUTTON BEEP ON' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_BUTTON_BEEP_ON

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_BUTTON_BEEP_OFF

Description Display 'BUTTON BEEP OFF' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_BUTTON_BEEP_OFF

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_NIGHTMODE_OFF

Description Display 'NIGHTMODE OFF' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_NIGHTMODE_OFF

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_NIGHTMODE_ON

Description Display 'NIGHTMODE ON' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_NIGHTMODE_ON

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_NIGHTMODE_AUTO

Description Display 'NIGHTMODE AUTO' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_NIGHTMODE_AUTO

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_CHIME_OFF

Description Display 'CHIME OFF' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_CHIME_OFF

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_CHIME_ON

Description Display 'CHIME ON' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_CHIME_ON

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_CHIME_AUTO

Description Display 'CHIME AUTO' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_CHIME_AUTO

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_ON_OFF_TIME

Description Display 'ON/OFF TIME' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_ON_OFF_TIME

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_PASSWORD_NEEDED

Description Display 'PASSWORD NEEDED' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_PASSWORD_NEEDED

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_SORTING_TRY_LATER

Description Display 'SORTING TRY LATER' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_SORTING_TRY_LATER

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_ENTER_PASSWORD

Description Display 'ENTER PASSWORD' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_ENTER_PASSWORD

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_PASSWORD_INVALID

Description Display 'PASSWORD INVALID' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_PASSWORD_INVALID

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_PASSWORD

Description Display 'PASSWORD' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_PASSWORD

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_NO_ENTRY_SELECTED

Description Display 'NO ENTRY SELECTED' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_NO_ENTRY_SELECTED

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_SNOOZE

Description Display 'SNOOZE' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_SNOOZE

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_END_OF_LIST

Description Display 'END OF LIST' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_END_OF_LIST

Assumptions None

Input None

Output None

Destroys BA, HL

Example

LCD_DISP_SMALL_DM_MSG_EDIT

Description Display 'EDIT' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_EDIT

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_ON_TIME

Description Display 'ON TIME' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_ON_TIME

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_OFF_TIME

Description Display 'OFF TIME' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_OFF_TIME

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_COMM_READY

Description Display 'COMM READY' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_COMM_READY

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_COMM_ERROR

Description Display 'COMM ERROR' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_COMM_ERROR

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SMALL_DM_MSG_PUSH_CROWN_IN

Description Display 'PUSH CROWN IN' on the LCD.

Usage LCD_DISP_SMALL_DM_MSG_PUSH_CROWN_IN

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_12HR

Description Display '12-HR' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_12HR

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_24HR

Description Display '24-HR' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_24HR

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_FREE

Description Display 'FREE' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_FREE

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_CHRONO

Description Display 'CHRONO' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_CHRONO

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_INT

Description Display 'INT' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_INT

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_HR_MIN

Description Display 'HR-MIN' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_HR_MIN

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_SECOND

Description Display 'SECOND' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_SECOND

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_TOTAL

Description Display 'TOTAL' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_TOTAL

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_DAILY

Description Display 'DAILY' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_DAILY

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_WKDAYS

Description Display 'WKDAYS' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_WKDAYS

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_WKENDS

Description Display 'WKENDS' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_WKENDS

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_WEEKLY

Description Display 'WEEKLY' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_WEEKLY

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_MNTHLY

Description Display 'MNTHLY' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_MNTHLY

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_YEARLY

Description Display 'YEARLY' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_YEARLY

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_ALARM

Description Display 'ALARM' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_ALARM

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_1_DAY

Description Display '1-DAY' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_1_DAY

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_MMDDYY

Description Display 'MM-DD-YY' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_MMDDYY

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_DDMMYY

Description Display 'DD.MM.YY' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_DDMMYY

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_YYMMDD

Description Display 'YY-MM-DD' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_YYMMDD

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_BATT

Description Display 'BATT' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_BATT

Assumptions None

Input None

Output None

Destroys BA, HL

Example

```
No Example
```

LCD_DISP_SEG_MSG_HOLD

Description Display 'HOLD' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_HOLD

Assumptions None

Input None

Output None

Destroys BA, HL

Example

```
No Example
```

LCD_DISP_SEG_MSG_STORE

Description Display 'STORE' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_STORE

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_TIME

Description Display 'TIME' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_TIME

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_1_MIN

Description Display '1 MIN' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_1_MIN

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_ALERT

Description Display 'ALERT' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_ALERT

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_SHOW

Description Display 'SHOW' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_SHOW

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_COUNT

Description Display 'COUNT' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_COUNT

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_SET

Description Display 'SET' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_SET

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_SEG_MSG_STOP

Description Display 'STOP' on the 9-segment area.

Usage LCD_DISP_SEG_MSG_STOP

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE1_SELECT

Description Display 'SELECT' on line 1 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE1_SELECT

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE1_ALARM

Description Display 'ALARM' on line 1 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE1_ALARM

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE2_ALARM

Description Display 'ALARM' on line 2 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE2_ALARM

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE1_ALARM_AT

Description Display 'ALARM AT' on line 1 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE1_ALARM_AT

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE2_TZ

Description Display 'TZ' on line 2 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE2_TZ

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE2_LAP

Description Display 'LAP' on line 2 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE2_LAP

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE2_LAPS

Description Display 'LAPS' on line 2 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE2_LAPS

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE1_MEMORY

Description Display 'MEMORY' in line 1 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE1_MEMORY

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE2_BEST_LAP

Description Display 'BEST LAP' in line 2 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE2_BEST_LAP

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE2_LAP_AVG

Description Display 'LAP AVG' in line 2 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE2_LAP_AVG

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE1_APPT_AT

Description Display 'APPT AT' in line 1 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE1_APPT_AT

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE1_HOURS

Description Display 'HOURS' in line 1 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE1_HOURS

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE1_MINS

Description Display 'MINS ' in line 1 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE1_MINS

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE2_PRIOR

Description Display 'PRIOR' in line 2 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE2_PRIOR

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE2_MINS

Description Display 'MINS ' in line 2 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE2_MINS

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE1_BDAY

Description Display 'BDAY' in line 1 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE1_BDAY

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE1_ANNV

Description Display 'ANNV' in line 1 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE1_ANNV

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE1_HOLIDAY

Description Display 'HOLIDAY' in line 1 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE1_HOLIDAY

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE1_VACATION

Description Display 'VACATION' in line 1 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE1_VACATION

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE1_4_DASHES

Description Display '----' in line 1 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE1_4_DASHES

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE1_CHIME

Description Display 'CHIME' in line 1 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE1_CHIME

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE2_ON

Description Display 'ON' in line 2 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE2_ON

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE2_OFF

Description Display 'OFF' in line 2 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE2_OFF

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE2_AUTO

Description Display 'AUTO' in line 2 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE2_AUTO

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE1_YEAR

Description Display 'YEAR' in line 1 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE1_YEAR

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE1_AM

Description Display 'AM' in line 1 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE1_AM

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE1_PM

Description Display 'PM' in line 1 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE1_PM

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE2_AM

Description Display 'AM' in line 2 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE2_AM

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

LCD_DISP_MAIN_DM_LINE2_PM

Description Display 'PM' in line 2 of the main DM area.

Usage LCD_DISP_MAIN_DM_LINE2_PM

Assumptions None

Input None

Output None

Destroys BA, HL

Example `No Example`

Utilities

UTL_DISPLAY_DAY_OF_WEEK

Description Displays the Day Of Week on the upper dot matrix region.

Usage UTL_DISPLAY_DAY_OF_WEEK

Assumptions None

Input A - Day of week to display

Index	Display Characters	Day of Week
0	SU	Sunday
1	MO	Monday
2	TU	Tuesday

3	WE	Wednesday
4	TH	Thursday
5	FR	Friday
6	SA	Saturday

Output None

Destroys None

Example

```
; Display SATURDAY (SA) in upper dot matrix
ld   A, #6
UTL_DISPLAY_DAY_OF_WEEK
```

UTL_DISPLAY_DATE_COMPLETE_AND_DOW

Description Displays the complete Date of an application based from the primary timezone configuration in the 6-digit segmented region, and displays the DOW in the upper dot matrix.

Usage UTL_DISPLAY_DATE_COMPLETE_AND_DOW

Assumptions Date Data Structure should be in the format: Day , Month, Year Lo, Year High, DOW.
All data is in BCD 2 digit per byte format. This means that :
"Day" is expressed as 0x01 to 0x31,
Month is 0x01 to 0x12,
etc.

Input IY - IYReg should point to the applications Day field.

Output None

Destroys BA, IX, HL

Example

```
; Display date and day of week information
ld   IY, #TODTimeZone1Day
UTL_DISPLAY_DATE_COMPLETE_AND_DOW
```

UTL_DISPLAY_DATE_COMPLETE

Description Displays the complete Date of an application based from the primary timezone configuration in the 6 digit segmented display

Usage UTL_DISPLAY_DATE_COMPLETE

Assumptions Date Data Structure should be in the format: Day , Month, Year Lo, Year High, DOW.
All data is in BCD 2 digit per byte format. This means that :
"Day" is expressed as 0x01 to 0x31,

Month is 0x01 to 0x12,
etc.

Input IY - IYReg should point to the applications Day field.

Output None

Destroys BA, IX, HL

Example
```
; Display date information in 6 digit 9 segment display.
ld   IY, #TODTimeZone1Day
UTL_DISPLAY_DATE_COMPLETE
```

UTL_DISPLAY_HR_MIN_DATA_L1

Description Displays a fixed-width hour and minute data on line 1.
Displays the colon between the hour and minute.
Displays either in 12 or 24 hour format. This will depend on the PTZ format.
Displays 'AM' or 'PM'. This will depend on the PTZ format.
Displays a fixed-width hour and mnute data on line 1.

Usage UTL_DISPLAY_HR_MIN_DATA_L1

Assumptions Data structure: Minute, Hour, where each number is stored as a single byte two digit BCD value.
Minutes is expressed as 0x00 to 0x59,
and Hours is expressed as 0x00 to 0x23

Input HL - address of the minute data.

Output None

Destroys IY

Example
```
; Display time information in Line 1 of the dot matrix
ld   HL, #TODTimeZone1Minutes
UTL_DISPLAY_HR_MIN_DATA_L1
```

UTL_DISPLAY_HR_MIN_DATA_L2

Description Displays a fixed-width hour and minute data on line 2.
Displays the colon between the hour and minute.
Displays either in 12 or 24 hour format. This will depend on the PTZ format.
Displays 'AM' or 'PM'. This will depend on the PTZ format.
Displays a fixed-width hour and mnute data on line 2

Usage UTL_DISPLAY_HR_MIN_DATA_L2

Assumptions Data structure: Minute, Hour, where each number is stored as a single byte two digit BCD value.
Minutes is expressed as 0x00 to 0x59,
and Hours is expressed as 0x00 to 0x23

Input HL - address of the minute data.

Output None

Destroys IY

Example

```
; Display time information in Line 2 of the dot matrix
ld   HL, #TODTimeZone1Minutes
UTL_DISPLAY_HR_MIN_DATA_L2
```

TIME ZONE RESOURCE API

The available resource index for the TOD resource are: 0x00, 0x01, 0x02, 0x03. Resource index 0x00 to 0x02 is reserved for the TOD application. Resource index 0x03 is reserved for the M851 OS.

This is the required time zone structure that is assumed for most Time Zone Resource APIs.

Offset	**Name**
0	KTODAPPINDEXOFFSET
1	KTODRESFLAGOFFSET
2	KTODUPDFLAGOFFSET
3	KTODSECONDOFFSET
4	KTODMINUTEOFFSET
5	KTODHOUROFFSET
6	KTODDATEOFFSET
7	KTODMONTHOFFSET
8	KTODYEARLOOFFSET
9	KTODYEARHIOFFSET
10	KTODDOWOFFSET
11	KTODWEEKOFFSET
12	KTODPREVIOUSSAMPLEOFFSET

KTODRESFLAGOFFSET Bit Definitions.

Bit	Name
0	bKReserved
1	bKActive
2	bKDispUpdRequestSec
3	bKDispUpdRequestMin
4	bKTODPrimaryTZ
5	bKTODEuroFormat
6	bKGeneratePopup
7	bKGenerateEvent

KTODUPDFLAGOFFSET Bit Definitions.

Bit	**Name**
0	bKTODSecondUpd
1	bKTODMinuteUpd
2	bKTODHourUpd
3	bKTODDateUpd
4	bKTODMonthUpd
5	bKTODYearUpd
6	bKTODWeekUpd

7	bKTODElapseMoreThan1Sec

Flag Manipulation

KTOD_ACTIVATE_RESOURCE

Description Enables a resource to be updated every second.

Usage KTOD_ACTIVATE_RESOURCE

Assumptions TOD resource must previously be owned by an application.

Input A - TOD Resource Index

Output None

Destroys None

Example
```
;enable the first TOD resource
ld  A, #0
KTOD_ACTIVATE_RESOURCE
```

KTOD_DEACTIVATE_RESOURCE

Description Disables updates to a specified resource.

Usage KTOD_DEACTIVATE_RESOURCE

Assumptions TOD resource must previously be owned by an application.

Input A - TOD Resource Index

Output None

Destroys None

Example
```
;prevent any updates to the TOD resource used as the primary time zone
ld  A, [COREPTZIndex]
KTOD_DEACTIVATE_RESOURCE
```

KTOD_ENABLE_DISP_UPD_SEC_EVENT

Description Send the event COREEVENT_DISPLAY_UPDATE_TODRES to the foreground application whenever seconds are updated in the specified resource.

Usage KTOD_ENABLE_DISP_UPD_SEC_EVENT

Assumptions TOD resource must previously be owned by an application.

Input A - TOD Resource Index

Output None

Destroys None

Example
```
;enable seconds updates from the primary time zone resource
ld  A, [COREPTZIndex]
KTOD_ENABLE_DISP_UPD_SEC_EVENT
```

KTOD_DISABLE_DISP_UPD_SEC_EVENT

Description Disable one second display updates to the foreground application.

Usage KTOD_DISABLE_DISP_UPD_SEC_EVENT

Assumptions TOD resource must previously be owned by an application.

Input A - TOD Resource Index

Output None

Destroys None

Example
```
;disable seconds updates from the primary time zone resource
ld  A, [COREPTZIndex]
KTOD_DISABLE_DISP_UPD_SEC_EVENT
```

KTOD_ENABLE_DISP_UPD_MIN_EVENT

Description Send the event COREEVENT_DISPLAY_UPDATE_TODRES to the foreground application whenever minutes are updated in the specified resource.

Usage KTOD_ENABLE_DISP_UPD_MIN_EVENT

Assumptions TOD resource must previously be owned by an application.

Input A - TOD Resource Index

Output None

Destroys None

Example
```
;enable minute update from the primary time zone resource
;this will cancel seconds update request
ld  A, [COREPTZIndex]
KTOD_ENABLE_DISP_UPD_MIN_EVENT
```

KTOD_DISABLE_DISP_UPD_MIN_EVENT

Description Disable one minute display updates to the foreground application.

Usage KTOD_DISABLE_DISP_UPD_MIN_EVENT

Assumptions TOD resource must previously be owned by an application.

Input A - TOD Resource Index

Output None

Destroys None

Example
```
;disable minute update from the primary time zone resource
ld  A, [COREPTZIndex]
KTOD_DISABLE_DISP_UPD_MIN_EVENT
```

KTOD_DEACTIVATE_ALL_DISPLAY_UPDATES

Description Disables all display updates of all active TOD resource from being passed to the application.

Usage KTOD_DEACTIVATE_ALL_DISPLAY_UPDATES

Assumptions TOD resource must previously be owned by an application.

Input None

Output None

Destroys None

Example
```
;disable both second and minute updates from the primary time zone resource
ld  A, [COREPTZIndex]
KTOD_DEACTIVATE_ALL_DISPLAY_UDPATES
```

KTOD_MAKE_AS_PRIMARY_TZ

Description Set the specified TOD resource as the primary time zone. The other active TOD resources are set as secondary time zone.

Usage KTOD_MAKE_AS_PRIMARY_TZ

Assumptions TOD resource must previously be owned by an application.

Input A - TOD Resource Index

Output None

Destroys None

Example
```
;make tod resource 2 as the primary time zone
ld  A, #2
KTOD_MAKE_AS_PRIMARY_TZ
```

KTOD_SET_FOR_EURO_FORMAT

Description Sets MONDAY as the first day of week used during week number calculation.

Usage KTOD_SET_FOR_EURO_FORMAT

Assumptions TOD resource must previously be owned by an application.

Input A - TOD Resource Index

Output None

Destroys None

Example
```
;set tod resource 1 to euro format
ld  A, #1
KTOD_SET_FOR_EURO_FORMAT
```

KTOD_SET_FOR_US_FORMAT

Description Sets SUNDAY as the first day of week used during week number calculation.

Usage KTOD_SET_FOR_US_FORMAT

Assumptions TOD resource must previously be owned by an application.

Input A - TOD Resource Index

Output None

Destroys None

Example
```
;set tod resource 1 to US format
ld  A, #1
KTOD_SET_FOR_US_FORMAT
```

Data Manipulation

KTOD_CALC_DOW

Description Calculates the day of the week from the given TOD structure.

Usage KTOD_CALC_DOW

Assumptions None

Input IY - Base address of the Day data in the structure

Output A - Day of the week

The system defines the following equates for Day of Week:

Constant	Value
SUNDAY	0
MONDAY	1
TUESDAY	2
WEDNESDAY	3
THURSDAY	4
FRIDAY	5
SATURDAY	6

Destroys B

Example
```
TODDateStructure:

   ;October 03, 2002
   db  03H    ;date in BCD
   db  10H    ;month in BCD
   db  02H    ;year lo in BCD
   db  20H    ;year hi in BCD

…

; compute the DOW from a given tod structure
```

```
ld  IY, #TODDateStructure
KTOD_CALC_DOW
```

KTOD_CALC_WEEK_NUMBER

Description Calculates the week number.

Usage KTOD_CALC_WEEK_NUMBER

Assumptions None

Input

IY	-	Base address of the Date data in the structure
ZeroFlag	-	1 = US Format, SUNDAY as first day. 0 = EURO Format, MONDAY as first day

Output

A	-	Week Number in BCD

Destroys B

Example

```
TODDateStructure:

   ;October 03, 2002
   db  03H    ;date in BCD
   db  10H    ;month in BCD
   db  02H    ;year lo in BCD
   db  20H    ;year hi in BCD

…

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; compute the week number from a given tod structure
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ld  A, [COREPTZFormat]
bit A, #bTODEuroFormat   ;sets ZeroFlag to correct value for EURO/US format
ld  IY, #TODDateStructure
KTOD_CALC_WEEK_NUMBER
```

KTOD_ADD_DAYS

Description Adds a number of days to the structure. This updates the month and day data only.

Usage KTOD_ADD_DAYS

Assumptions None

Input

A	-	The number of days to add in BCD format.
IY	-	Base address of DAY in the TOD structure.

Output

B.bKTODMonthUpd	-	0 - if month data didn't change 1 – month data was updated
B. bKTODYearUpd		0 - if year data didn't change 1 – year data was updated

Destroys A

Example

```
TODDateStructure:

   ;October 28, 2002
   db  28H    ;date in BCD
   db  10H    ;month in BCD
   db  02H    ;year lo in BCD
   db  20H    ;year hi in BCD

NumberOfDaysToAdd:

   db  07H    ;7 days

…

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; compute the week number from a given tod structure
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ld  A, [NumberOfDaysToAdd]
ld  IY, #TODDateStructure
KTOD_ADD_DAYS

;results to TODDataStructure to contain: November 5, 2002
;results to B.kTODMonthUpd = 1  (month update detected)
;results to B.KTODYearUpd = 0   (no year update required)

;check if we need to increase the year data
…
```

KTOD_SUBTRACT_DAYS

Description Subtracts a number of days from the structure. This will only update the month and day data in the data structure.

Usage KTOD_SUBTRACT_DAYS

Assumptions None

Input

A	-	The number of days to subtract in BCD format.
IY	-	Base address of DAY in the TOD structure.

Output

B.bKTODMonthUpd	-	0 - if month data didn't change 1 – month data was updated
B.bKTODYearUpd		0 - if year data didn't change 1 – year data was updated

Destroys A

Example

```
TODDateStructure:

   ;October 03, 2002
   db  28H    ;date in BCD
   db  10H    ;month in BCD
   db  02H    ;year lo in BCD
   db  20H    ;year hi in BCD
   db  04H    ;DOW (Thursday)

…


; subtract 7 days from the specified TOD date structure
ld  IY, #TODDateStructure
ld  A, #7
KTOD_SUBTRACT_DAYS

;results to TODDataStructure to contain: September 27, 2002
;results to B.kTODMonthUpd = 1  (month update detected)
;results to B.KTODYearUpd = 0   (no year update required)

; If it is required to completely adjust the year and DOW information to
; compensate for year rollovers (DEC TO JAN) and leap year,
; then following code sections are required.

; adjust the month and year to compensate for leap year and year rollover
; the API below requires the B register to contain the month and year update
; flags set correctly.  In this example, the B register is setup by the
; KTOD_SUBTRACT_DAYS API
ld   IY, #TODDateStructure
KTOD_ADJUST_DATE_AND_YEAR

; compute the new DOW for the specified TOD structure and store it back
; into the date structure
ld   IY, #TODDateStructure
KTOD_CALC_DOW
ld   [TODDateStructure+4], A
```

KTOD_ADD_YEARS

Description Adds a number of years to the structure.

Usage KTOD_ADD_YEARS

Assumptions None

Input

A	-	The number of year to add in BCD format.
IY	-	Base address of YearLo data in TOD structure

Output None

Destroys None

Example

```
TODDateStructure:

   ;October 28, 2002
   db  28H    ;date in BCD
   db  10H    ;month in BCD
   db  02H    ;year lo in BCD
   db  20H    ;year hi in BCD

NumberOfDaysToAdd:

   db  07H    ;7 days

…

   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   ; compute the week number from a given tod structure
   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

   ld  A, [NumberOfDaysToAdd]
   ld  IY, #TODDateStructure
   KTOD_ADD_DAYS

   ;check if we need to increase the year data
   bit B, # KTODYearUpd
   jr  Z, DoNotIncrementTheYear

   ; add one year since we have detected a december to January rollover
   ld  A, #1                       ; add 1 year
   ld  IY, #(TODDateStructure+2)   ; point to the year lo address
   KTOD_ADD_YEARS

DoNotIncrementTheYear:
```

KTOD_SUBTRACT_YEARS

Description Subtracts a number of years from the structure

Usage KTOD_SUBTRACT_YEARS

Assumptions None

Input

A	-	The number of year to subtract in BCD format.
IY	-	Base address of YearLo data in TOD structure

Output None

Destroys None

Example

```
TODDateStructure:

   ;October 28, 2002
   db  28H    ;date in BCD
   db  10H    ;month in BCD
   db  02H    ;year lo in BCD
   db  20H    ;year hi in BCD
```

```
…
   ; subtract one year
   ld  A, #1                        ; subtract 1 year
   ld  IY, #(TODDateStructure+2)    ; point to the year lo address
   KTOD_SUBTRACT_YEARS
```

KTOD_CORRECT_MDY_DATA

Description Checks the date, month and year data of the structure if it is valid. If it is invalid, then it is set back to a known correct value which is its maximum possible date for a month.

Adjust the year data to '6999' if its not within the boundary conditions: 0001 - 6999

Adjust the month data to 'DECEMBER' if its not within the boundary conditions: JANUARY – DECEMBER.

Adjust the day data based on the month and year information with compensation for leap year.

Usage KTOD_CORRECT_MDY_DATA

Assumptions None

Input IY - Base address of Date data in TOD structure
Date:Month:YearLo:YearHi

Output None

Destroys None

Example

```
TODDateStructure:

   ;October 28, 2002
   db  28H    ;date in BCD
   db  10H    ;month in BCD
   db  02H    ;year lo in BCD
   db  20H    ;year hi in BCD
…
   ; adjust to correct MDY
   ld  IY, #TODDateStructure   ; point to the date
   KTOD_CORRECT_MDY_DATA
```

KTOD_ADJUST_DATE_AND_YEAR

Description If year is updated, subtract 1 from the year data. If month is updated, subtract 1 from day data if the month is FEBRUARY and not a leap year.

This can only be called if there is a date subtraction.

Usage KTOD_ADJUST_DATE_AND_YEAR

Assumptions None

Input
IY - Base address of Day data in TOD structure
B - Contains the year and month update flags

Output None

Destroys None

Example

```
TODDateStructure:

   ;October 03, 2002
   db  28H    ;date in BCD
   db  10H    ;month in BCD
   db  02H    ;year lo in BCD
   db  20H    ;year hi in BCD

NumberOfDaysToSubtract:

   db  07H    ;7 days

…

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; compute the week number from a given tod structure
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ld  A, [NumberOfDaysToSubtract]
ld  IY, #TODDateStructure
KTOD_SUBTRACT_DAYS

;results to TODDataStructure to contain: September 27, 2002
;results to B.kTODMonthUpd = 1  (month update detected)
;results to B.KTODYearUpd = 0   (no year update required)

; adjust the year and month and date due to the subtract operation
ld  IY, # TODDateStructure
KTOD_ADJUST_DATE_AND_YEAR
```

Data Transfer

KTOD_COPY_RESOURCE_TO_RESOURCE

Description Copies the tod resource data from into another tod resource structure. App index, resource flag, update flag are not copied.

Usage KTOD_COPY_RESOURCE_TO_RESOURCE

Assumptions None

Input
A - Resource Index of source structure
B - Resource Index of destination structure

Output None

Destroys None

Example
```
;copy the primary time zone data to tod resource #3
ld  A, [COREPTZIndex]
ld  B, #03H
KTOD_COPY_RESOURCE_TO_RESOURCE
```

KTOD_COPY_TIME_MIN_TO_DOW_FROM_RESOURCE

Description Copies day of week, year, month, date, hour and minute data of a specified resource into a memory buffer.

Usage KTOD_COPY_TIME_MIN_TO_DOW_FROM_RESOURCE

Assumptions The destination buffer is allocated with the number of bytes that will be copied.

Input
A - Resource Index of source structure
IY - Start address of the destination buffer.

Output None

Destroys None

Example
```
MyWorkBuffer:

  db 00   ;minute
  db 00   ;hour
  db 00   ;date
  db 00   ;month
  db 00   ;year lo
  db 00   ;year hi
  db 00   ;dow

…

;copy the primary time zone data to work buffer
ld  A, [COREPTZIndex]
ld  IY, #MyWorkBuffer
KTOD_COPY_TIME_MIN_TO_DOW_FROM_RESOURCE
```

KTOD_COPY_TIME_FROM_RESOURCE

Description Copies week number, day of week, year, month, date, hour, minute and second data of a specified

resource into a buffer.

Usage KTOD_COPY_TIME_FROM_RESOURCE

Assumptions The destination buffer is allocated with the number of bytes that will be copied.

Input

A	-	Resource Index of source structure
IY	-	Start address of the destination buffer.

Output None

Destroys None

Example

```
MyWorkBuffer:

  db 00   ;second
  db 00   ;minute
  db 00   ;hour
  db 00   ;date
  db 00   ;month
  db 00   ;year lo
  db 00   ;year hi
  db 00   ;dow
  db 00   ;week number
…

;copy the primary time zone data to work buffer
ld  A, [COREPTZIndex]
ld  IY, #MyWorkBuffer
KTOD_COPY_TIME_FROM_RESOURCE
```

KTOD_COPY_MDY_FROM_RESOURCE

Description Copies year, month and date data of a specified resource into a buffer.

Usage KTOD_COPY_MDY_FROM_RESOURCE

Assumptions The destination buffer is allocated with the number of bytes that will be copied.

Input

A	-	Resource Index of source structure
IY	-	Start address of the destination buffer.

Output None

Destroys None

Example

```
MyWorkBuffer:

  db 00   ;date
  db 00   ;month
  db 00   ;year lo
  db 00   ;year hi
```

```
…

;copy the primary time zone data to work buffer
ld  A, [COREPTZIndex]
ld  IY, #MyWorkBuffer
KTOD_COPY_MDY_FROM_RESOURCE
```

KTOD_COPY_HMS_FROM_RESOURCE

Description Copies hour, minute and second data of a specified resource into a buffer.

Usage KTOD_COPY_HMS_FROM_RESOURCE

Assumptions The destination buffer is allocated with the number of bytes that will be copied.

Input

A	-	Resource Index of source structure
IY	-	Start address of the destination buffer.

Output None

Destroys None

Example

```
MyWorkBuffer:

  db 00   ;second
  db 00   ;minute
  db 00   ;hour

…

;copy the primary time zone data to work buffer
ld  A, [COREPTZIndex]
ld  IY, #MyWorkBuffer
KTOD_COPY_HMS_FROM_RESOURCE
```

KTOD_WRITE_TIME_TO_RESOURCE

Description Copies year, month, date, hour, minute and second data from a buffer into a specified resource.

Copies the data from the KTODPreviousRTCSample and use it as the previous sample.
Corrects the validity of date data stored, calculates the day of the week and week number based on the Euro Format bit. If the Euro Format bit is going to be changed, then changed it first before writing the data to the resource.

Usage KTOD_WRITE_TIME_TO_RESOURCE

Assumptions None

Input
A - Resource Index of destination structure
IY - Start address of the source buffer.

Output None

Destroys None

Example

```
MyWorkBuffer:

  ; 13:30:00 May 14, 2002
  db 00H   ;second
  db 30H   ;minute
  db 13H   ;hour
  db 14H   ;date
  db 05H   ;month
  db 02H   ;year lo
  db 20H   ;year hi

…

;load new time data to resource
ld  A, [COREPTZIndex]
ld  IY, #MyWorkBuffer
KTOD_WRITE_TIME_TO_RESOURCE
```

KTOD_WRITE_MDY_TO_RESOURCE

Description Copies year, month and date data from a buffer into a specified resource.

This also corrects the validity of date data stored, calculates the day of the week and week number based on the Euro Format bit. If the Euro Format bit is going to be changed, then changed it first before writing the data to the resource.

Usage KTOD_WRITE_MDY_TO_RESOURCE

Assumptions None

Input
A - Resource Index of destination structure
IY - Start address of the source buffer.

Output None

Destroys None

Example

```
MyWorkBuffer:

  ; 13:30:00 May 14, 2002
  db 00H   ;second
  db 30H   ;minute
  db 13H   ;hour
  db 14H   ;date
  db 05H   ;month
  db 02H   ;year lo
  db 20H   ;year hi
```

```
…

;load new month date year data to resource
ld  A, [COREPTZIndex]
ld  IY, #(MyWorkBuffer+3)      ;point to date
KTOD_WRITE_MDY_TO_RESOURCE
```

KTOD_WRITE_HMS_TO_RESOURCE

Description Copies hour, minute and second data from a buffer into a specified resource

Usage KTOD_WRITE_HMS_TO_RESOURCE

Assumptions None

Input

A	-	Resource Index of destination structure
IY	-	Start address of the source buffer.

Output None

Destroys None

Example

```
MyWorkBuffer:

  ; 13:30:00 May 14, 2002
  db 00H   ;second
  db 30H   ;minute
  db 13H   ;hour
  db 14H   ;date
  db 05H   ;month
  db 02H   ;year lo
  db 20H   ;year hi

…

;load new hour minute second data to resource
ld  A, [COREPTZIndex]
ld  IY, #(MyWorkBuffer+0)      ;point to second
KTOD_WRITE_HMS_TO_RESOURCE
```

Resource Utilities

KTOD_LOAD_RESOURCE_START_ADDRESS

Description Loads the start address of a specified resource into HL

Usage KTOD_LOAD_RESOURCE_START_ADDRESS

Assumptions None

Input A - Resource index of resource

Output HL - Base address of the resource

Destroys None

Example
```
;get the base address of the primary time zone resource
ld  A, [COREPTZIndex]
KTOD_LOAD_RESOURCE_START_ADDRESS
```

KTOD_GET_UPDATE_FLAGS

Description Gets the update flags of a specified tod resource.

Usage KTOD_GET_UPDATE_FLAGS

Assumptions None

Input A - Resource index of resource

Output A - Update Flags

Update Flag bit structure is shown below:

Bit Position	Flag Name
0	X – don’t care
1	bKTODMinuteUpd
2	bKTODHourUpd
3	bKTODDateUpd
4	bKTODMonthUpd
5	bKTODYearUpd
6	bKTODWeekUpd
7	X – don’t care

Destroys None

Example
```
;get the current update status flag of the primary time zone resource
ld  A, [COREPTZIndex]
KTOD_GET_UPDATE_FLAGS
```

KTOD_CHECK_IF_LEAP_YEAR

Description Checks if a specified year is a leap year

Usage KTOD_CHECK_IF_LEAP_YEAR

Assumptions None

Input BA - YearHi:YearLo in BCD.

Output bZeroFlag - TRUE -- leap year
FALSE -- not a leap year

Destroys A

Example

```
MyWorkBuffer:

  ; 13:30:00 May 14, 2002
  db 00H   ;second
  db 30H   ;minute
  db 13H   ;hour
  db 14H   ;date
  db 05H   ;month
  db 02H   ;year lo
  db 20H   ;year hi

…

;check if current year is a leap year
ld  BA, [MyWorkBuffer+5]   ;load the year into BA register
KTOD_CHECK_IF_LEAP_YEAR
jr  Z, ItIsALeapYear
jr  NZ, ItIsNotALeapYear
```

KTOD_GET_MAX_DAYS_OF_A_MONTH

Description Returns the maximum number of days in the specified month.

Usage KTOD_GET_MAX_DAYS_OF_A_MONTH

Assumptions None

Input L - Month in BCD

Output B - Maximum number of days in the month.

Destroys L

Example

```
MyWorkBuffer:

  ; 13:30:00 May 14, 2002
  db 00H   ;second
  db 30H   ;minute
  db 13H   ;hour
  db 14H   ;date
  db 05H   ;month
```

```
  db 02H   ;year lo
  db 20H   ;year hi

…

;check if current year is a leap year
ld  L, [MyWorkBuffer+4]   ;load the month into L register
KTOD_GET_MAX_DAYS_OF_A_MONTH

;B register will contain the max days for the queried month
```

KTOD_DIVIDE_BY_2

Description Divides a 2-byte BCD structure by two.

Usage KTOD_DIVIDE_BY_2

Assumptions None

Input BA - BCD Structure (Hi:Lo)

Output BA - Quotient

Destroys None

Example

```
;divide BCD number by 2
ld  BA,#1250H
KTOD_DIVIDE_BY_2

;Results: BA = 0625H
```

KTOD_CONVERT_BCD_TO_HEX

Description Converts a 2-byte BCD structure into a two-byte HEX structure.

Usage KTOD_CONVERT_BCD_TO_HEX

Assumptions None

Input BA - BCD Structure (Hi:Lo)

Output BA - HEX Structure (Hi:Lo)

Destroys IY, HL

Example

```
;convert BCD number to hex
ld  BA,#1250H
KTOD_CONVERT_BCD_TO_HEX
```

```
;Results: BA = 04E2H  OR BA = 1250 (DECIMAL)
```

KTOD_CHECK_IF_DIVISIBLE_BY_4

Description Checks a BCD number if it is divisible by 4. This is only used to check if year is a leap year since the watch only support the year 2000 to 2099.

Usage KTOD_CHECK_IF_DIVISIBLE_BY_4

Assumptions None

Input A - BCD Number

Output ZeroFlag - 1 -- divisible by 4
0 -- not divisible by 4

Destroys A

Example

```
MyWorkBuffer:

  ; 13:30:00 May 14, 2002
  db 00H   ;second
  db 30H   ;minute
  db 13H   ;hour
  db 14H   ;date
  db 05H   ;month
  db 02H   ;year lo
  db 20H   ;year hi

;check if year lo is divisible by 4
ld  A,[MyWorkBuffer+5]
KTOD_CHECK_IF_DIVISIBLE_BY_4
jr  NZ, ValueNotDivisibleBy4
jr  Z, ValueDivisibleBy4
```

KTOD_GET_PTZ_ADDRESS

Description Returns the primary time zone address for the given input offset.

Usage KTOD_GET_PTZ_ADDRESS

Assumptions None

Input B - Offset into the TOD resource

Output IY - Address in the Primary TOD resource data structure for the given input offset

Destroys None

Example
```
;Get the absolute address of the seconds data of primary time zone
ld  B, # KTODSECONDOFFSET
KTOD_GET_PTZ_ADDRESS

;store 0 to primary time zone seconds
ld  A, #00H
Ld  [IY], A
```

KTOD_GET_TZ_ADDRESS

Description Returns the time zone resourceaddress for the given resource index and input offset.

Usage KTOD_GET_PTZ_ADDRESS

Assumptions None

Input
- A - Resource index
- B - Offset into the TOD resource

Output
- IY - Address in the Primary TOD resource data structure for the given input offset

Destroys None

Example
```
;Get the absolute address of the seconds data of TOD resource
ld  B, # KTODSECONDOFFSET
KTOD_GET_TZ_ADDRESS

;store 0 to tod resource seconds
ld  A, #00H
Ld  [IY], A
```

TIME ZONE CHECK RESOURCE API

The available resource index for the TZC resource are: 0x00, 0x01, 0x02, 0x03, 0x04, 0x05.

Offset	Name
0	KTZCAPPINDEXOFFSET
1	KTZCRESFLAGOFFSET
2	KTZCUPDFLAGOFFSET
3	KTZCMINUTEOFFSET
4	KTZCHOUROFFSET
5	KTZCDATEOFFSET
6	KTZCMONTHOFFSET
7	KTZCYEARLOOFFSET
8	KTZCYEARHIOFFSET

KTZCRESFLAGOFFSET Bit Definitions.

Bit	Name
0	bKReserved
1	bKActive
2	*Unused*
3	*Unused*
4	*Unused*
5	*Unused*
6	bKGeneratePopup
7	bKGenerateEvent

KTZCUPDFLAGOFFSET Bit Definitions.

Bit	Name
0	*Unused*
1	*Unused*
2	*Unused*
3	bKTZCDateUpd
4	bKTZCMonthUpd
5	bKTZCYearUpd
6	*Unused*
7	*Unused*

Flag Manipulation

KTZC_DEACTIVATE_ALL_RESOURCES

Description Disables all periodic checking on all the time zone check resource.

Usage KTZC_DEACTIVATE_ALL_RESOURCES

Assumptions None

Input None

Output None

Destroys None

Example
```
;user just entered tod setting
;do not allow any checking to the time zone check while changing
;the primary time zone
KTZ_DEACTIVATE_ALL_RESOURCES
```

KTZC_ACTIVATE_RESOURCE

Description Activates the specified time zone check resource.

Usage KTZC_ACTIVATE_RESOURCE

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output None

Destroys None

Example
```
;appointment application just found the next occurrence of appointment
…

;loaded the occurrence data into resource
…

;activate the resource.  First get the resource index owned by application
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTTZCRESOURCEOFFSET0]
KTZC_ACTIVATE_RESOURCE
```

KTZC_DISABLE_RESOURCE

Description Deactivates the specified time zone check resource.

Usage KTZC_DISABLE_RESOURCE

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output None

Destroys None

Example
```
;deactivate the resource owned by the current application
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTTZCRESOURCEOFFSET0]
KTZC_DEACTIVATE_RESOURCE
```

KTZC_SETUP_POPUP_GENERATION

Description Setup resource to generate popup on matching resource with time zone.

Usage KTZC_SETUP_POPUP_GENERATION

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output None

Destroys None

Example
```
;specify that a popup be generated
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTTZCRESOURCEOFFSET0]
KTZC_SETUP_POPUP_GENERATION
```

KTZC_SETUP_EVENT_GENERATION

Description Setup resource to generate event on matching resource with time zone.

Usage KTZC_SETUP_EVENT_GENERATION

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output None

Destroys None

Example
```
;specify that An event be generated and passed to the foreground application
ld  IY, [CORECurrentASDAddress]
```

```
ld  A, [IY + APPTTZCRESOURCEOFFSET0]
KTZC_SETUP_EVENT_GENERATION
```

KTZC_CANCEL_POPUPEVENT_GENERATION

Description Cancel popup or event generation of specified resource

Usage KTZC_CANCEL_POPUPEVENT_GENERATION

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output None

Destroys None

Example
```
; do not generate any event or popup
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTTZCRESOURCEOFFSET0]
KTZC_CANCEL_POPUPEVENT_GENERATION
```

Data Manipulation

KTZC_RETURN_RESOURCE_OWNER

Description Returns the application index that owns the specified resource index

Usage KTZC_RETURN_RESOURCE_OWNER

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output B - Application Index

Destroys None

Example
```
; return the owner of the time zone check resource 0
ld  A, #0
KTZC_RETURN_RESOURCE_OWNER
```

KTZC_SETUP_MDY_CHECK

Description Specify if the resource should check the month, day and year data stored in the resource.

Usage KTZC_SETUP_MDY_CHECK

Assumptions No check is being done if the specified resource is already allocated

Input

A - Resource index
B - MDY check data pattern

The following MDY bit patterns can be ORed together to form the criteria for checking month, day and year.

Bit Position	Flag Name	Description
0	unused	
1	unused	
2	unused	
3	bKTZCDateUpd	1 - Check date data
4	bKTZCMonthUpd	1 - Check month data
5	bKTZCYearUpd	1 - Check year data
6	unused	
7	unused	

Output None

Destroys None

Example

```
; we want to check only the Date data in the TZC resource for an
; automatic monthly appointment popup
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTTZCRESOURCEOFFSET0]
ld  B, #( bKTZCDateUpd )
KTZC_SETUP_MDY_CHECK
```

KTZC_SETUP_HOUR_MINUTE_MDY

Description Update the hour, minute and MDY data structure with user specified data.

Usage KTZC_SETUP_HOUR_MINUTE_MDY

Assumptions No check is being done if the specified resource is already allocated

Input

A - Resource index
IY - Source Address of Hour, Minute, MDY data structure.

The following is the required TZC structure:

Offset	Description
0	Minute data

1	Hour data
2	Date date
3	Month data
4	Year Lo data
5	Year Hi data

Output None

Destroys None

Example

```
NextOccuringAppointment:

  db  00H   ;minute
  db  00H   ;hour
  db  00H   ;date
  db  00H   ;month
  db  00H   ;yearlo
  db  00H   ;yearhi

; setup the time and date to generate the appointment popup
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTTZCRESOURCEOFFSET0]
ld  IY, # NextOccuringAppointment
KTZC_SETUP_HOUR_MINUTE_MDY
```

KTZC_SETUP_HOUR_MINUTE

Description Update the hour and minute data structure with user specified data.

Usage KTZC_SETUP_HOUR_MINUTE

Assumptions No check is being done if the specified resource is already allocated

Input

A	-	Resource index
IY	-	Source Address of Hour, Minute.

The following is the required TZC structure:

Offset	Description
0	Minute data
1	Hour data

Output None

Destroys None

Example

```
NextOccuringAppointment:

  db  00H   ;minute
  db  00H   ;hour
  db  00H   ;date
```

```
  db  00H   ;month
  db  00H   ;yearlo
  db  00H   ;yearhi

; setup the hour and minute
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTTZCRESOURCEOFFSET0]
ld  IY, #(NextOccuringAppointment+0)   ;point to minute
KTZC_SETUP_HOUR_MINUTE
```

KTZC_SETUP_MONTH

Description Update the month data structure with user specified data.

Usage KTZC_SETUP_MONTH

Assumptions No check is being done if the specified resource is already allocated

Input

A	-	Resource index
IY	-	Source Address of Month data

The following is the required TZC structure:

Offset	Description
0	Month data

Output None

Destroys None

Example

```
NextOccuringAppointment:

  db  00H   ;minute
  db  00H   ;hour
  db  00H   ;date
  db  00H   ;month
  db  00H   ;yearlo
  db  00H   ;yearhi

; setup the month
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTTZCRESOURCEOFFSET0]
ld  IY, #(NextOccuringAppointment+3)  ;point to month
KTZC_SETUP_MONTH
```

KTZC_SETUP_DATE

Description Update the date data structure with user specified data.

Usage KTZC_SETUP_DATE

Assumptions No check is being done if the specified resource is already allocated

Input

A	-	Resource index
IY	-	Source Address of Date data

The following is the required TZC structure:

Offset	Description
0	Date data

Output None

Destroys None

Example

```
NextOccuringAppointment:

  db  00H   ;minute
  db  00H   ;hour
  db  00H   ;date
  db  00H   ;month
  db  00H   ;yearlo
  db  00H   ;yearhi

; setup the month
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTTZCRESOURCEOFFSET0]
ld  IY, #(NextOccuringAppointment+2)  ;point to date
KTZC_SETUP_MONTH
```

KTZC_SETUP_YEAR

Description Update the year data structure with user specified data.

Usage KTZC_SETUP_YEAR

Assumptions No check is being done if the specified resource is already allocated

Input

A	-	Resource index
IY	-	Source Address of Year data structure

The following is the required TZC structure:

Offset	Description
0	Year Lo data
1	Year Hi data

Output None

Destroys None

Example

```
NextOccuringAppointment:

  db  00H   ;minute
  db  00H   ;hour
  db  00H   ;date
  db  00H   ;month
  db  00H   ;yearlo
  db  00H   ;yearhi

; setup the month
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTTZCRESOURCEOFFSET0]
ld  IY, #(NextOccuringAppointment+4)  ;point to year lo
KTZC_SETUP_YEAR
```

BACKUP RESOURCE API

The available resource index for the BCK resource are: 0x00, 0x01, 0x02.

Offset	Name
0	KBCKAPPINDEXOFFSET
1	KBCKRESFLAGOFFSET
2	KBCKSNOOZEOFFSET

KBCKRESFLAGOFFSET Bit Definitions.

Bit	Name
0	bKReserved
1	bKActive
2	*Unused*
3	*Unused*
4	*Unused*
5	*Unused*
6	bKGeneratePopup
7	bKGenerateEvent

Flag Manipulation

KBCK_DEACTIVATE_ALL_RESOURCES

Description Disables all periodic checking on all the backup resource.

Usage KBCK_DEACTIVATE_ALL_RESOURCES

Assumptions None

Input None

Output None

Destroys None

Example

KBCK_ACTIVATE_RESOURCE

Description Allows update every minute to the snooze data in the resource

Usage KBCK_ACTIVATE_RESOURCE

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output None

Destroys None

Example
```
; activate resource
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTBCKRESOURCEOFFSET0]
KBCK_ACTIVATE_RESOURCE
```

KBCK_DEACTIVATE_RESOURCE

Description Disallows updates every minute to the snooze data in the resource.

Usage KBCK_DEACTIVATE_RESOURCE

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output None

Destroys None

Example
```
; deactivate resource
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTBCKRESOURCEOFFSET0]
KBCK_DEACTIVATE_RESOURCE
```

KBCK_SETUP_POPUP_GENERATION

Description Setup resource to generate popup when snooze time expires.

Usage KBCK_SETUP_POPUP_GENERATION

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output None

Destroys None

Example

```
; setup for popup
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTBCKRESOURCEOFFSET0]
KBCK_SETUP_POPUP_GENERATION
```

KBCK_SETUP_EVENT_GENERATION

Description Setup resource to generate event when snooze time expires.

Usage KBCK_SETUP_EVENT_GENERATION

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output None

Destroys None

Example

```
; setup for event to be passed to foreground application
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTBCKRESOURCEOFFSET0]
KBCK_SETUP_EVENT_GENERATION
```

KBCK_CANCEL_POPUPEVENT_GENERATION

Description Cancel popup or event generation of specified resource

Usage KBCK_CANCEL_POPUPEVENT_GENERATION

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output None

Destroys None

Example

```
; do not generate an event or popup
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTBCKRESOURCEOFFSET0]
KBCK_CANCEL_POPUPEVENT_GENERATION
```

Data Manipulation

KBCK_RETURN_RESOURCE_OWNER

Description Returns the application index that owns the specified resource index.

Usage KBCK_RETURN_RESOURCE_OWNER

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output B - Application Index

Destroys None

Example
```
;get owner of the backup resource #1
ld  A, #1
KBCK_RETURN_RESOURCE_OWNER
```

KBCK_SETUP_SNOOZE_TIME

Description Sets up the snooze time of the specified resource index.
Cancels any queued backup check resource of the same index.

Usage KBCK_SETUP_SNOOZE_TIME

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index
B - Snooze time specified in HEX.

Output None

Destroys None

Example
```
; setup the snooze time for 10 minutes
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTBCKRESOURCEOFFSET0]
ld  B, #10
KBCK_SETUP_SNOOZE_TIME
```

STOPWATCH RESOURCE API

The available resource index for the STP resource are: 0x00, 0x01.

Offset	Name
0	KSTPAPPINDEXOFFSET
1	KSTPRESFLAGOFFSET
2	KSTPUPDFLAGOFFSET
3	KSTPPREVHUNDTHSSAMPLEOFFSET
4	KSTPPREVSECONDSSAMPLEOFFSET
5	KSTPHUNDREDTHSOFFSET
6	KSTPSECONDOFFSET
7	KSTPMINUTEOFFSET
8	KSTPHOUROFFSET
9	KSTPTMRIDOFFSET
10	KSTPPOPUPSOURCEAPPIDOFFSET
11	KSTPSTPIDOFFSET

KSTPRESFLAGOFFSET Bit Definitions.

Bit	Name
0	bKReserved
1	bKActive
2	bKDispUpdRequest
3	bKTMRLink
4	bKSTPLink
5	bKNotReset
6	bKGeneratePopup
7	bKGenerateEvent

KSTPUPDFLAGOFFSET Bit Definitions.

Bit	Name
0	bKSTPSecondUpd
1	bKSTPMinuteUpd
2	bKSTPHourUpd
3	*Unused*
4	bKSTPSecondNonZero
5	bKSTPMinuteNonZero
6	bKSTPHoursNonZero
7	bKSTPRunout

Data Manipulation

KSTP_LINK_NEXT_RESOURCE

Description Indicates that the next resource is a link to this specified resource.

Usage KSTP_LINK_NEXT_RESOURCE

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output None

Destroys HL

Example
```
;request that when the current resource is started, it will also
;start the next resource (by index)
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTSTPRESOURCEOFFSET0]
KSTP_LINK_NEXT_RESOURCE
```

KSTP_ENABLE_DISP_UPD_EVENT

Description Allows the foreground application to receive display update events from the specified resource.

Usage KSTP_ENABLE_DISP_UPD_EVENT

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output None

Destroys HL

Example
```
;request that any updates done to the stopwatch at 16HZ interval that
;a event be passed to the foreground application for display updates
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTSTPRESOURCEOFFSET0]
KSTP_ENABLE_DISP_UPD_EVENT
```

KSTP_DISABLE_DISP_UPD_EVENT

Description Cancels any requests to send the foreground application to receive display update events from the specified resource.

Usage KSTP_DISABLE_DISP_UPD_EVENT

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output None

Destroys HL

Example
```
;do not sent any display update events when stopwatch resource is
;updated
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTSTPRESOURCEOFFSET0]
KSTP_DISABLE_DISP_UPD_EVENT
```

KSTP_DEACTIVATE_ALL_DISPLAY_UPDATES

Description Cancels all requests of all stopwatch resource to send the foreground application to receive display update events.

Usage KSTP_DEACTIVATE_ALL_DISPLAY_UPDATES

Assumptions No check is being done if the specified resource is already allocated

Input None

Output None

Destroys HL, B

Example
```
;disable all stopwatch resource display update event generation
;to the foreground application
KSTP_DEACTIVATE_ALL_DISPLAY_UPDATES
```

KSTP_SETUP_POPUP_GENERATION

Description Setup the specified resource to generate popup on stopwatch runout.

Usage KSTP_SETUP_POPUP_GENERATION

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output None

Destroys HL

Example
```
;request a popup when the stopwatch resource has reached 100 hours
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTSTPRESOURCEOFFSET0]
KSTP_SETUP_POPUP_GENERATION
```

KSTP_SETUP_EVENT_GENERATION

Description Setup resource to generate event on stopwatch runout.

Usage KSTP_SETUP_EVENT_GENERATION

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output None

Destroys HL

Example
```
;request an event to be passed to the foreground application
;when the stopwatch resource has reached 100 hours
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTSTPRESOURCEOFFSET0]
KSTP_SETUP_EVENT_GENERATION
```

KSTP_CANCEL_POPUPEVENT_GENERATION

Description Cancel popup or event generation of specified resource.

Usage KSTP_CANCEL_POPUPEVENT_GENERATION

Assumptions No check is being done if the specified resource is already allocated

Input A - Resource index

Output None

Destroys HL

Example
```
;no events when the stopwatch has reached 100 hours.
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTSTPRESOURCEOFFSET0]
KSTP_CANCEL_POPUPEVENT_GENERATION
```

KSTP_START_RESOURCE

Description Starts a specific stopwatch resource

Usage KSTP_START_RESOURCE

Assumptions KRESStopwatchHundthsDataBuffer and KRESStopwatchSecondsDataBuffer holds the value of the free running counter snapshot or the values where this resource should use as reference to start.

Input A - Resource index

Output None

Destroys None

Example
```
;start stopped stopwatch resource
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTSTPRESOURCEOFFSET0]
KSTP_START_RESOURCE
```

KSTP_STOP_RESOURCE

Description Stops a specific stopwatch resource.

Usage KSTP_STOP_RESOURCE

Assumptions KRESStopwatchHundthsDataBuffer and KRESStopwatchSecondsDataBuffer holds the value of the free running counter snapshot or the values where this resource should use as reference to stop.

Input A - Resource index

Output None

Destroys None

Example
```
;stop a running stopwatch resource
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTSTPRESOURCEOFFSET0]
KSTP_STOP_RESOURCE
```

KSTP_STOP_ALL_RESOURCES

Description Stops all active stopwatch resource.

Usage KSTP_STOP_ALL_RESOURCES

Assumptions KRESStopwatchHundthsDataBuffer and KRESStopwatchSecondsDataBuffer holds the value of the free running counter snapshot or the values where this resource should use as reference to stop.

Input None

Output None

Destroys None

Example
```
;stop all stopwatch resource
KSTP_STOP_ALL_RESOURCES
```

KSTP_TAKE_SPLIT

Description Updates a specific stopwatch resource so a split time can be stored.

Usage KSTP_TAKE_SPLIT

Assumptions KRESStopwatchHundthsDataBuffer and KRESStopwatchSecondsDataBuffer holds the value of the free running counter snapshot or the values where this resource should use as reference to stop.

Input A - Resource index

Output None

Destroys None

Example
```
;take a split and freeze updates to the display (resource is still running)
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTSTPRESOURCEOFFSET0]
KSTP_TAKE_SPLIT
KSTP_DISABLE_DISP_UPD_EVENT
;display the stopwatch data for user to see
…
```

KSTP_RESET_RESOURCE

Description Resets a specific stopwatch resource to 00:00:00.00.

Usage KSTP_RESET_RESOURCE

Assumptions None

Input A - Resource index

Output None

Destroys None

Example

```
;check if STOP switch was depress
cp  A, #COREEVENT_STOPRESETDEPRESS
jr  NZ, check_next_event1

;reset the stopwatch
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTSTPRESOURCEOFFSET0]
KSTP_RESET_RESOURCE
ret

check_next_event1:
```

KSTP_RESET_ALL_RESOURCE

Description Resets all stopwatch resource to 00:00:00.00.

Usage KSTP_RESET_ALL_RESOURCE

Assumptions None

Input None

Output None

Destroys A

Example

```
;reset all stopwatch resource
KSTP_RESET_ALL_RESOURCE
```

KSTP_CLEAR_RESOURCE_DATA

Description Resets a specific stopwatch resource to 00:00:00.00.

Usage KSTP_CLEAR_RESOURCE_DATA

Assumptions None

Input A - Resource index

Output None

Destroys HL

Example

```
;clear the lap stopwatch resource since we just took a split time
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTSTPRESOURCEOFFSET1]
KSTP_CLEAR_RESOURCE_DATA
```

Data Transfer

KSTP_COPY_RESOURCE_TO_BUFFER

Description Copies stopwatch resource data H:M:S.h data from a specified resource index into a buffer.

Usage KSTP_COPY_RESOURCE_TO_BUFFER

Assumptions None

Input

A	-	Resource index
IY	-	Destination address to store hundredths, second, minute and hour in BCD format.

Output None

Destroys HL

Example

```
MyStopwatchBuffer:

  db   00H   ;hundredths
  db   00H   ;seconds
  db   00H   ;minute
  db   00H   ;hour
…
;copy stopwatch data to buffer
ld  IY, [CORECurrentASDAddress]
ld  A, [IY + APPTSTPRESOURCEOFFSET0]
ld  IY, #MyStopwatchBuffer
KSTP_COPY_RESOURCE_TO_BUFFER
```

Resource Utilities

KSTP_COMPARE_BUFFER_FROM_RESOURCE

Description Compares a buffer to a resource

Usage KSTP_COMPARE_BUFFER_FROM_RESOURCE

Assumptions None

Input A - Resource index
IY - Hours Information of the Buffer using a stopwatch data structure

Output Set/Clears the ZeroFlag and Carry Flag based on the following conditions:

If Resource == Buffer
ZeroFlag == TRUE, CarryFlag == FALSE

If Resource < Buffer
ZeroFlag == FALSE, CarryFlag == TRUE

If Resource > Buffer
ZeroFlag == FALSE, CarryFlag == FALSE

Destroys A, B, IY, HL

Example

```
MyStopwatchBuffer:

  db   00H  ;app index
  db   00H   ;resource flag
  db   00H   ;update flag
  db   00H   ;previous hundredth sample
  db   00H   ;previous seconds sample
  db   00H   ;hundredths
  db   00H   ;seconds
  db   00H   ;minute
  db   00H   ;hour
  db   00H   ;timer id
  db   00H   ;popupsource app id
  db   00H   ;stopwatch id
…
;compare stopwatch date with buffer data
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTSTPRESOURCEOFFSET0]
ld  IY, #[MyStopwatchBuffer+8]   ;point to hour data
KSTP_COMPARE_BUFFER_FROM_RESOURCE

jr  Z,  ResourceIsEqualBufferData
jr  C,  ResourceIsLessThanBufferData
jr  NC, ResourceIsgreaterThanBufferData
```

KSTP_GET_RESOURCE_STATUS

Description Gets the status of a specified resource index.

Usage KSTP_GET_RESOURCE_STATUS

Assumptions None

Input A - Resource index

Output AReg holds the status of the resource. The following values are returned in A to indicate the status oif

the resource.

Value	Resource Status
0x00	Reset
0x20	Not reset and stopped.
0x22	Not reset and running

Destroys HL

Example

```
;get resource status
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTSTPRESOURCEOFFSET0]
KSTP_GET_RESOURCE_STATUS
cp  A, #00H
jr  Z, StopwatchResourceIsReset
cp  A, #20H
jr  Z, StopwatchNotResetAndStopped
cp  A, #22H
jr  Z, StopwatchNotResetAndRunning
```

TIMER RESOURCE API

The available resource index for the TMR resource are: 0x00, 0x01, 0x02.

Offset	Name
0	KTMRAPPINDEXOFFSET
1	KTMRRESFLAGOFFSET
2	KTMRUPDFLAGOFFSET
3	KTMRPREVHUNDTHSSAMPLEOFFSET
4	KTMRPREVSECONDSSAMPLEOFFSET
5	KTMRWORKHUNDREDTHSOFFSET
6	KTMRWORKSECONDOFFSET
7	KTMRWORKMINUTEOFFSET
8	KTMRWORKHOUROFFSET
9	KTMRTMRIDOFFSET
10	KTMRSTPIDOFFSET
12	KTMRUSERSECONDOFFSET
13	KTMRUSERMINUTEOFFSET
14	KTMRUSERHOUROFFSET
15	KTMRPREWARNSECONDOFFSET
16	KTMRPREWARNMINUTEOFFSET
17	KTMRPREWARNHOUROFFSET
18	KTMRCOUNTEROFFSET
19	KTMRNUMOFREPSOFFSET

KTMRRESFLAGOFFSET Bit Definitions.

Bit	Name
0	bKReserved
1	bKActive
2	bKDispUpdRequest
3	bKTMRLink
4	bKSTPLink
5	bKNotReset
6	bKGeneratePopup
7	bKGenerateEvent

KTMRUPDFLAGOFFSET Bit Definitions.

Bit	Name
0	bKTMRSecondUpd
1	bKTMRMinuteUpd
2	bKTMRHourUpd
3	bKTMRPreWarnEnTracking
4	bKTMRPreWarnReqTracking

```
5          bKTMRRepeat
6          bKTMRDirection
7          Unused
```

Data Manipulation

KTMR_SETUP_USER_AND_WORK_HMS

Description Sets up the user set and working HMS data.

When setup for a count-down operation, user data is the initial data for countdown and will be loaded into the working time structure when reset.

When setup for a count-up operation, the working data usually starts at 0:00.00.00 (using the reset API) and counts up until the value stored in the user time structure. At runout, it will stop the timer and generated the required event for background task processing or popup processing.

The working time variables can be loaded with other data through the SETUP_WORKING_HMS API.

Usage KTMR_SETUP_USER_AND_WORK_HMS

Assumptions The counting direction has already been set.

Input

A	-	Resource index
IY	-	Points to the beginning of the Seconds, Minutes and Hour structure from memory buffer

Output None

Destroys None

Example

```
NextUserSetSecond:

   ;user time: 01:30:00
   db  00H  ;second
   db  30H  ;minute
   db  01H  ;hour
…

; setup both user and working time structure in a countdown operation
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
ld  IY, #NextUserSetSecond
KTMR_SETUP_USER_AND_WORK_HMS
```

KTMR_SETUP_WORKING_HMS

Description Sets up the working HMS data from the user data.

Usage KTMR_SETUP_WORKING_HMS

Assumptions The counting direction has already been set.

Input A - Resource index

Output None

Destroys None

Example

```
;setup working time structure from user time structure
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
KTMR_SETUP_WORKING_HMS
```

KTMR_SETUP_PREWARNING_HMS

Description Sets up the pre-warning data. If pre-warning data is equal to the user set data then the pre-warning tracking will not be enabled. If it is successful then pre-warning tracking will be enabled.

Usage None

Assumptions The counting direction has already been set.

Input A - Resource index
IY - Points to the beginning of the Seconds, Minutes and Hour structure from memory buffer

Output None

Destroys None

Example

```
NextUserSetSecond:

   ;user time: 01:30:00
   db  00H  ;second
   db  30H  ;minute
   db  01H  ;hour

PreNotificationTime:

   ;based on half of user time: 01:30:00/2 = 00:45:00
   db  00H  ;second
   db  45H  ;minute
   db  00H  ;hour

…

;setup prewarning time structure
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
ld  IY, # PreNotificationTime
KTMR_SETUP_PREWARNING_HMS
```

KTMR_DISABLE_PREWARNING_TRACKING

Description Disables the pre-warning processing by the specified resource.

Usage KTMR_DISABLE_PREWARNING_TRACKING

Assumptions None

Input A - Resource index

Output None

Destroys None

Example
```
;disable prewarning tracking
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
KTMR_DISABLE_PREWARNING_TRACKING
```

KTMR_ENABLE_DISP_UPD_EVENT

Description Enable display update event to be passed to the foreground application.

Usage None

Assumptions The counting direction has already been set.

Input A - Resource index

Output None

Destroys B

Example
```
;enable display update events to be passed to foreground application
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
KTMR_ENABLE_DISP_UPD_EVENT
```

KTMR_DISABLE_DISP_UPD_EVENT

Description Disable display update event to be passed to the foreground application.

Usage KTMR_DISABLE_DISP_UPD_EVENT

Assumptions None

Input A - Resource index

Output None

Destroys B

Example
```
;disable display update events to be passed to foreground application
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
KTMR_DISABLE_DISP_UPD_EVENT
```

KTMR_DEACTIVATE_ALL_DISPLAY_UPDATES

Description Deactivates display update for all timer resources.

Usage KTMR_DEACTIVATE_ALL_DISPLAY_UPDATES

Assumptions None

Input None

Output None

Destroys B

Example
```
;cancel all display update events on all timer resource
KTMR_DEACTIVATE_ALL_DISPLAY_UPDATES
```

KTMR_SETUP_COUNTUP

Description Set the timer resource as a count up resource.

Usage KTMR_SETUP_COUNTUP

Assumptions None

Input A - Resource index

Output None

Destroys B

Example
```
;setup the timer resource for count UP operation
ld  IX, [CORECurrentASDAddress]
```

```
ld  A, [IX + APPTMRRESOURCEOFFSET0]
KTMR_SETUP_COUNTUP
```

KTMR_SETUP_COUNTDOWN

Description Make the timer resource as a count down resource.

Usage KTMR_SETUP_COUNTDOWN

Assumptions None

Input A - Resource index

Output None

Destroys B

Example

```
;setup the timer resource for count DOWN operation
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
KTMR_SETUP_COUNTDOWN
```

KTMR_SET_RUNNING_REPEAT_COUNTER

Description Forcibly change the value of the repeat counter.

Usage KTMR_SET_RUNNING_REPEAT_COUNTER

Assumptions None

Input A - Resource index
B - Starting repetition counter formatted as HEX data.

Output None

Destroys None

Example

```
;change the current value of the repeat counter
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
ld  B, #0                              ;set repeat counter to 0
KTMR_SET_RUNNING_REPEAT_COUNTER
```

KTMR_GET_RUNNING_REPEAT_COUNTER

Description Get the number of repetitions of a specified resource.

Usage KTMR_GET_RUNNING_REPEAT_COUNTER

Assumptions None

Input A - Resource index

Output B - Running repeat counter stored as a HEX value.

Destroys HL

Example
```
;get current running counter
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
KTMR_GET_RUNNING_REPEAT_COUNTER
```

KTMR_SETUP_POPUP_GENERATION

Description Set the bit that tells the resource that it has to generate a popup when the resource expires.

Usage KTMR_SETUP_POPUP_GENERATION

Assumptions None

Input A - Resource index

Output None

Destroys B

Example
```
;request a popup when counter has reached 0 (if countdown) or
;if reach specified max time (if countup)
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
KTMR_SETUP_POPUP_GENERATION
```

KTMR_SETUP_EVENT_GENERATION

Description Set the bit that tells the resource that it has to generate an event when the timer expires.

Usage KTMR_SETUP_EVENT_GENERATION

Assumptions None

Input A - Resource index

Output None

Destroys B

Example
```
;request an event to be passed to the foreground application when
;counter has reached 0 (if countdown) or if reach specified max
;time (if countup)
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
KTMR_SETUP_EVENT_GENERATION
```

KTMR_CANCEL_POPUP_AND_EVENT_GENERATION

Description Cancel popup event and event generation.

Usage KTMR_CANCEL_POPUP_AND_EVENT_GENERATION

Assumptions None

Input A - Resource index

Output None

Destroys None

Example
```
;do note generate a popup or event when timer expires
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
KTMR_CANCEL_POPUP_AND_EVENT_GENERATION
```

KTMR_SETUP_REPEAT_AT_END

Description Enable the repeat at end action and sets up the number of times we need to repeat the timer.

Usage KTMR_SETUP_REPEAT_AT_END

Assumptions None

Input A - Resource index
B - Number of Repetitions formatted as HEX data.

If number of repetitions is 0 then it is a continuous repeat.

Output None

Destroys None

Example
```
;setup timer to repeat at end
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
KTMR_SETUP_REPEAT_AT_END
```

KTMR_CANCEL_REPEAT_AT_END

Description Disables the repeat at end action.

Usage KTMR_CANCEL_REPEAT_AT_END

Assumptions None

Input A - Resource index

Output None

Destroys B

Example
```
;cancel timer to repeat at end
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
KTMR_CANCEL_REPEAT_AT_END
```

KTMR_SETUP_TMR_LINK

Description Set a timer link on timer ending.

Usage KTMR_SETUP_TMR_LINK

Assumptions None

Input A - Timer Resource index to establish the link from
B - Timer Resource index to link to when the Timer ends

Output None

Destroys None

Example
```
;setup the timer resource to start when the specified timer expires
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
```

```
ld  B, [IX + APPTMRRESOURCEOFFSET1]
KTMR_SETUP_TMR_LINK
```

KTMR_SETUP_STP_LINK

Description Set the bit that indicates to link a stopwatch when the timer expires. The stopwatch resource to start will be started when the timer expires.

Usage KTMR_SETUP_STP_LINK

Assumptions None

Input A - Timer resource to link Stopwatch Resource to

Output None

Destroys None

Example

```
;setup the timer resource to start a stopwatch resource (that is reset)
;when the specified timer expires
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
KTMR_SETUP_TMR_LINK
```

KTMR_CANCEL_ALL_LINKS

Description Cancels all links to TMR and STP of a specified resource.

Usage KTMR_CANCEL_ALL_LINKS

Assumptions None

Input A - Resource index

Output None

Destroys None

Example

```
;cancel all timer and stopwatch resource links when timer expires
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
KTMR_CANCEL_ALL_LINKS
```

KTMR_RESET_RESOURCE

Description Sets up the working HMS data depending on the direction of the timer. Clears the running repeat counter. This also stops the total and stoppage time if timer caused a synchro resource to start.

Usage KTMR_RESET_RESOURCE

Assumptions None

Input A - Resource index

Output None

Destroys None

Example
```
;reset the timer resource
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
KTMR_RESET_RESOURCE
```

KTMR_START_RESOURCE

Description Start background update for a timer resource with the stopwatch data buffers already loaded with the value to be used.

A synchro resource will be started from reset if the specified timer resource is started from reset. The application owner of this timer will also be the one to control the synchro resource. The synchro stoppage time will be stopped if this resource caused it to start.

Usage KTMR_START_RESOURCE

Assumptions None

Input A - Resource index

Output None

Destroys None

Example
```
;start the timer resource
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
KTMR_START_RESOURCE
```

KTMR_STOP_RESOURCE

Description Stop background update for a timer resource with the stopwatch data buffers already loaded with the value to be used.

The synchro resource stoppage time will be started if the specified timer resource caused it to start from reset. The synchro running time will be stopped.

Usage KTMR_STOP_RESOURCE

Assumptions None

Input A - Resource index

Output None

Destroys None

Example
```
;stop a running timer resource
ld  IX, [CORECurrentASDAddress]
ld  A, [IX + APPTMRRESOURCEOFFSET0]
KTMR_STOP_RESOURCE
```

Data Transfer

KTMR_COPY_RESOURCE_TO_BUFFER

Description Copy the working HMS data to a buffer of a specified resource using the time resource structure for hour, minute and seconds.

Usage KTMR_COPY_RESOURCE_TO_BUFFER

Assumptions None

Input A - Resource index
IY - Start address of buffer to store the timer resource data.

Output None

Destroys HL, IY

Example
```
TimerTimeBuffer:

   db   00H   ;second
   db   00H   ;minute
   db   00H   ;hour
…

;copy working timer hour, minute and seconds to buffer
ld  IX, [CORECurrentASDAddress]
ld  A,  [IX + APPTMRRESOURCEOFFSET0]
ld  IY, #TimerTimeBuffer
KTMR_COPY_RESOURCE_TO_BUFFER
```

Resource Utilities

KTMR_GET_STATUS

Description Gets the status of a specified resource index.

Usage KTMR_GET_STATUS

Assumptions None

Input A - Resource index

Output B - Resource status

The following constants indicate the status of the timer resource.

Constant
KTMRSTATUSRESET
KTMRSTATUSSTOPPED
KTMRSTATUSRUNNING

Destroys HL

Example

```
ld  IX, [CORECurrentASDAddress]
ld  A,  [IX + APPTMRRESOURCEOFFSET0]
KTMR_GET_STATUS

cp  B, #KTMRSTATUSRESET
jr  Z, TimerIsReset

cp  B, #KTMRSTATUSSTOPPED
jr  Z, TimerIsNotResetAndStopped

cp  B, #KTMRSTATUSRUNNING
jr  Z, TimerIsNotResetAndRunning
```

KTMR_GET_UPDATE_BITS

Description Get the update bits of a specified resource.

Usage KTMR_GET_UPDATE_BITS

Assumptions None

Input A - Resource index

Output B - Update flag

The following bit flags indicate which timer data has been updated.

Bit Flag	Description
bKTMRSecondUpd	Set to 1 if seconds data has been updated
bKTMRMinuteUpd	Set to 1 if minute data has been updated
bKTMRHourUpd	Set to 1 if hour data has been updated

Destroys HL

Example

```
   ld  IX, [CORECurrentASDAddress]
   ld  A,  [IX + APPTMRRESOURCEOFFSET0]
   KTMR_GET_UPDATE_BITS

   bit  B, #bKTMRHourUpd
   jr   Z, HourMinuteSecondDisplay

   bit  B, #bKTMRMinuteUpd
   jr   Z, MinuteSecondDisplay

   bit  B, #bKTMRSecondUpd
   jr   Z, SecondDisplay

HourMinuteSecondDisplay:

   ;display hour data

MinuteSecondDisplay:

   ;display minute data

SecondDisplay:

   ;display second data
```

SYNCHRO RESOURCE API

The available resource index for the STP resource are: 0x00.

Offset	Name
0	KSYNAPPINDEXOFFSET
1	KSYNRESFLAGOFFSET
2	KSYNAPPOWNEROFFSET
3	KSYNTOTALPREVHUNDSAMPLEOFFSET
4	KSYNTOTALPREVSECONDSAMPLEOFFSET
5	KSYNTOTALHUNDREDTHSOFFSET
6	KSYNTOTALSECONDOFFSET
7	KSYNTOTALMINUTEOFFSET
8	KSYNTOTALHOUROFFSET
9	KSYNUPDFLAGTOTALOFFSET
10	KSYNSTOPPREVHUNDSAMPLEOFFSET
11	KSYNSTOPPREVSECONDSAMPLEOFFSET
12	KSYNSTOPHUNDREDTHSOFFSET
13	KSYNSTOPSECONDOFFSET
14	KSYNSTOPMINUTEOFFSET
15	KSYNSTOPHOUROFFSET
16	KSYNUPDFLAGSTOPPAGEOFFSET

KSYNRESFLAGOFFSET Bit Definitions.

Bit	Name
0	bKReserved
1	bKActive
2	bKDispUpdRequest
3	bKSYNStartStoppage
4	*Unused*
5	bKNotReset
6	bKGeneratePopup
7	bKGenerateEvent

KSYNUPDFLAGTOTALOFFSET Bit Definitions.

Bit	Name
0	bKSYNTotalSecondUpd
1	bKSYNTotalMinuteUpd
2	bKSYNTotalHourUpd
3	*Unused*
4	bKSYNTotalSecondNonZero
5	bKSYNTotalMinuteNonZero
6	bKSYNTotalHourNonZero

7	bKSYNTotalRunout

KSYNUPDFLAGSTOPPAGEOFFSET Bit Definitions.

Bit	Name
0	bKSYNStoppageSecondUpd
1	bKSYNStoppageMinuteUpd
2	bKSYNStoppageHourUpd
3	*Unused*
4	bKSYNStoppageSecondNonZero
5	bKSYNStoppageMinuteNonZero
6	bKSYNStoppageHourNonZero
7	bKSYNStoppageRunout

Data Manipulation

KSYN_GET_CAUSED_TO_START

Description Returns the application index that caused the specified synchro resource to start.

Usage KSYN_GET_CAUSED_TO_START

Assumptions None

Input None

Output A - Application Index

Destroys HL

Example
```
;get the application index that caused the synchro resource to start
KSYN_GET_CAUSED_TO_START
```

KSYN_GET_TOTAL_TIME_UPD_BITS

Description Gets the TOTAL TIME update status.

Usage KSYN_GET_TOTAL_TIME_UPD_BITS

Assumptions None

Input None

Output A - Update status for the total time structure

The following bit flags indicate which data field has been updated.

Bit Flag	Description
bKSYNTotalSecondUpd	TRUE, if second data is updated
bKSYNTotalMinuteUpd	TRUE, if minute data is updated
bKSYNTotalHourUpd	TRUE, if hour data is updated
bKSYNTotalSecondNonZero	TRUE, if seconds is not zero
bKSYNTotalMinuteNonZero	TRUE, if minutes is not zero
bKSYNTotalHourNonZero	TRUE, if hours is not zero
bKSYNTotalRunout	TRUE, if synchro total time has ran out

Destroys HL

Example

```
;get the total time update flags
KSYNC_GET_TOTAL_TIME_UPD_BITS
bit  A,  #bKSYNCTotalSecondUpd
jr   NZ, SynchroTotalTimeSecondUpdated
bit  A,  #bKSYNCTotalMintueUpd
jr   NZ, SynchroTotalTimeMinuteUpdated
bit  A,  #bKSYNTotalRunout
jr   NZ, SynchroTotalHasRunout
```

KSYN_GET_STOPPAGE_TIME_UPD_BITS

Description Gets the STOPPAGE TIME update status

Usage KSYN_GET_STOPPAGE_TIME_UPD_BITS

Assumptions None

Input None

Output A - Update status for the stoppage time structure

The following bit flags indicate which data field has been updated.

Bit Flag	Description
bKSYNStoppageSecondUpd	TRUE, if second data is updated
bKSYNStoppageMinuteUpd	TRUE, if minute data is updated
bKSYNStoppageHourUpd	TRUE, if hour data is updated
bKSYNStoppageSecondNonZero	TRUE, if seconds is not zero
bKSYNStoppageMinuteNonZero	TRUE, if minutes is not zero
bKSYNStoppageHourNonZero	TRUE, if hours is not zero
bKSYNStoppageRunout	TRUE, if synchro stoppage time has ran out

Destroys HL

Example

```
;get the stoppage time update flags
KSYNC_GET_STOPPAGE_TIME_UPD_BITS
bit  A,  #bKSYNCStoppageSecondUpd
jr   NZ, SynchroStoppageTimeSecondUpdated
bit  A,  #bKSYNCStoppageMintueUpd
jr   NZ, SynchroStoppageTimeMinuteUpdated
bit  A,  #bKSYNStoppageRunout
```

```
jr   NZ, SynchroTotalHasRunout
```

KSYN_GET_TOTAL_STATUS

Description Gets the status of the total time. Either reset, stop or running.

Usage KSYN_GET_TOTAL_STATUS

Assumptions None

Input None

Output B - Status of total time operations.

The following constants indicate the status of total time operations.

Constant	Description
KSYNTOTALSTATUSRESET	Reset
KSYNTOTALSTATUSSTOPPED	Stop
KSYNTOTALSTATUSRUNNING	Running

Destroys HL

Example

```
;get current status of the total time
KSYN_GET_TOTAL_STATUS

cp  B, #KSYNTOTALSTATUSRESET
jr  Z, SynchroTotalTimeIsReset

cp  B, #KSYNTOTALSTATUSSTOPPED
jr  Z, SynchroTotalTimeIsNotResetAndStopped

cp  B, #KSYNTOTALSTATUSRUNNING
jr  Z, SynchroTotalTimeIsNotResetAndRunning
```

KSYN_GET_STOPPAGE_STATUS

Description Gets the status of the stoppage time. Either reset, stop or running.

Usage KSYN_GET_STOPPAGE_STATUS

Assumptions None

Input None

Output B - Status of stoppage time operations.

The following constants indicate the status of stoppage time operations.

Constant	Description
KSYNSTOPSTATUSRESET	Reset
KSYNSTOPSTATUSSTOPPED	Stop
KSYNSTOPSTATUSRUNNING	Running

Destroys HL

Example

```
;get current status of the stoppage time
KSYN_GET_STOPPAGE_STATUS

cp  B, #KSYNSTOPSTATUSRESET
jr  Z, SynchroStoppageTimeIsReset

cp  B, #KSYNSTOPSTATUSSTOPPED
jr  Z, SynchroStoppageTimeIsNotResetAndStopped

cp  B, #KSYNSTOPSTATUSRUNNING
jr  Z, SynchroStoppageTimeIsNotResetAndRunning
```

KSYN_ENABLE_DISP_UPD_EVENT

Description Allows the resource to pass a display update event to the foreground application.

Usage KSYN_ENABLE_DISP_UPD_EVENT

Assumptions None

Input None

Output None

Destroys HL

Example

```
;enable display events
KSYN_ENABLE_DISP_UPD_EVENT
```

KSYN_DISABLE_DISP_UPD_EVENT

Description Disallows the resource to pass a display update event to the foreground application.

Usage KSYN_DISABLE_DISP_UPD_EVENT

Assumptions None

Input None

Output None

Destroys HL

Example
```
;disable display events
KSYN_DISABLE_DISP_UPD_EVENT
```

KSYN_SETUP_POPUP_GENERATION

Description Setup resource to generate popup when synchro time has ran-out.

Usage KSYN_SETUP_POPUP_GENERATION

Assumptions None

Input None

Output None

Destroys HL

Example
```
;request popup when synchro time has reached 100 hours
KSYN_SETUP_POPUP_GENERATION
```

KSYN_SETUP_EVENT_GENERATION

Description Setup resource to generate event when synchro time has ran-out

Usage KSYN_SETUP_EVENT_GENERATION

Assumptions None

Input None

Output None

Destroys HL

Example
```
;request event to be passed to foreground application when
;synchro time has reached 100 hours
KSYN_SETUP_EVENT_GENERATION
```

KSYN_CANCEL_POPUPEVENT_GENERATION

Description Cancel resource to generate popup and event when synchro time has ran-out.

Usage KSYN_CANCEL_POPUPEVENT_GENERATION

Assumptions None

Input None

Output None

Destroys HL

Example
```
;no event or popup when synchro time has reached 100 hours
KSYN_CANCEL_POPUPEVENT_GENERATION
```

KSYN_RESET_AND_START

Description Resets and starts the Synchro timer.

Usage KSYN_RESET_AND_START

Assumptions KRESStopwatchHundthsDataBuffer and KRESStopwatchSecondsDataBuffer contains the snap shot of the free running hundredths and seconds counter.

Input None

Output None

Destroys None

Example
```
;reset and start synchro timer
KSYN_RESET_AND_START
```

KSYN_STOP_TOTAL_AND_STOPPAGE_TIME

Description Stops synchro total and stoppage time and update the values with the stored RTC.

Usage KSYN_STOP_TOTAL_AND_STOPPAGE_TIME

Assumptions KRESStopwatchHundthsDataBuffer and KRESStopwatchSecondsDataBuffer contains the snap shot of the free running hundredths and seconds counter.

This will only be called when the resource is forcibly stopped due to switch depression (in synchro mode).

Input None

Output None

Destroys None

Example
```
;stop both total time and stoppage time
KSYN_STOP_TOTAL_AND_STOPPAGE_TIME
```

KSYN_RESET_RESOURCE

Description Resets a synchro resource to 00:00:00.00.

Usage KSYN_RESET_RESOURCE

Assumptions None

Input None

Output None

Destroys None

Example
```
;reset synchro resource to 0
KSYN_RESET_RESOURCE
```

Data Transfer

KSYN_COPY_TOTAL_TO_BUFFER

Description Copies the total time H:M:S.h into a buffer

Usage KSYN_COPY_TOTAL_TO_BUFFER

Assumptions None

Input HL - Start address of buffer to write into

Output HL - Points to the next byte after the destinations structure

Destroys B, IY

Example

```
MyTimeBuffer:

   db   00H   ;hundredth
   db   00H   ;second
   db   00H   ;minute
   db   00H   ;hour

…

   ld  HL, #MyTimeBuffer
   KSYN_COPY_TOTAL_TO_BUFFER
```

KSYN_COPY_STOPPAGE_TO_BUFFER

Description Copies the stoppage time H:M:S.h into a buffer.

Usage KSYN_COPY_STOPPAGE_TO_BUFFER

Assumptions None

Input HL - Start address of buffer to write into

Output HL - Points to the next byte after the destinations structure

Destroys B, IY

Example

```
MyTimeBuffer:

   db   00H   ;hundredth
   db   00H   ;second
   db   00H   ;minute
   db   00H   ;hour

…

   ld  HL, #MyTimeBuffer
   KSYN_COPY_STOPPAGE_TO_BUFFER
```

UTILITIES API

KRES_CLEAR_N_BYTES

Description Clears n consecutive bytes

Usage KRES_CLEAR_N_BYTES

Assumptions None

Input

B	-	number of bytes to clear
HL		start address of the buffer to be cleared

Output None

Destroys B, HL

Example

```
MyBuffer:
    db   0FFH
    db   0ECH
    db   012H
    db   034H

MyBufferEnd:

…

;setup the number of bytes to clear
ld  B, #(MyBufferEnd - MyBuffer)

;setup the start address to clear
ld  HL, #MyBuffer
KRES_CLEAR_N_BYTES
```

HW_RESET_MCU

Description Reset the MCU.

Usage HW_RESET_MCU

Assumptions None

Input None

Output None

Destroys Doesn’t matter.

Example

```
;reset the microcontroller
HW_RESET_MCU
```

HW_RESET_WATCHDOG

Description Reset the WATCHDOG timer to prevent a reset.

Normal operation of the M851 OS will reset the watchdog timer. If a certain operation within the wristapp takes longer than 3 seconds, then it should use the API within the procedure to prevent the watch reset.

Usage HW_RESET_WATCHDOG

Assumptions BR is point to the hardware memory page

Input None

Output None

Destroys None

Example
```
;reset the watchdog timer
HW_RESET_WATCHDOG
```

HW_LAMP_POPUP_REQUEST

Description Sets lamp during popup request bit to enable lamp flashing during a popup.

This allows the audio generator to turn on the lamp during the OFF phases (read: no frequency generated or a rest tone) of the melody pattern.

Usage HW_LAMP_POPUP_REQUEST

Assumptions None

Input None

Output None

Destroys HL

Example
```
;request to alternate between lamp and buzzer during a popup melody
HW_LAMP_POPUP_REQUEST
AUDSTART_SYSTEM_MELODY AUDALARMMELODY, AUDSENDMELODYDONEEVENT
```

HW_LAMP_POPUP_REQUEST_OFF

Description Clears lamp during popup request bit to disable lamp flashing during a popup.

Usage HW_LAMP_POPUP_REQUEST_OFF

Assumptions None

Input None

Output None

Destroys HL

Example

```
;popup is complete request for a popdown
HW_LAMP_POPUP_REQUEST_OFF
ORE_REQ_POPDOWN
```

UTL_BINARY_MATH_MODE

Description Setup micro-processor to perform add and sub instructions in binary mode.

By default, the MCU is in binary mode during WristApp execution. The MCU will be in binary mode until the macro UTL_DECIMAL_MATH_MODE is called. On return to the M851 OS, the system will set it back to binary mode.

Usage UTL_BINARY_MATH_MODE

Assumptions None

Input None

Output None

Destroys None

Example

```
UTL_BINARY_MATH_MODE
ld  A, #25H
add A, #09H

;result A = 2EH
```

UTL_DECIMAL_MATH_MODE

Description Setup micro-processor to perform add and sub instructions in binary mode. NOTE: Not all add and sub

instructions support decimal operations.

By default, the MCU is in binary mode during WristApp execution. The MCU will be in decimal mode until the macro UTL_BINARY_MATH_MODE is called. On return to the M851 OS, the system will set it back to binary mode.

Usage UTL_DECIMAL_MATH_MODE

Assumptions None

Input None

Output None

Destroys None

Example
```
UTL_DECIMAL_MATH_MODE
ld  A, #25H
add A, #09H

;result A = 34H
```

Conversion

UTL_CONVERT_HEX_TO_3DIGIT_BCD

Description Converts a 1-byte HEX number to 2-byte BCD number.

Usage UTL_CONVERT_HEX_TO_3DIGIT_BCD

Assumptions None

Input
L - hex data to be converted (0x00 – 0xFF)

Output
B - 100's digit BCD data
A - 10's digit BCD data (high nibble)
1's digit BCD data (low nibble)

Destroys None

Example
```
;convert hex 0FFH to BCD
ld  L, #FFH
UTL_CONVERT_HEX_TO_3DIGIT_BCD

;result: B = 02H
;result: A = 55H
```

UTL_CONVERT_HEX_TO_2DIGIT_BCD

Description Converts a 1-byte HEX number to 1-byte BCD number.

Usage UTL_CONVERT_HEX_TO_2DIGIT_BCD

Assumptions None

Input L - hex data to be converted (0x00 – 0x63)

Maximum HEX number that can be converted is '0x63' since the 1-byte BCD can only hold up to '99'.

Output A - 10's digit BCD data (high nibble)
1's digit BCD data (low nibble)

Destroys None

Example

```
;convert HEX to BCD
ld  L, #63H
UTL_CONVERT_HEX_TO_2DIGIT_BCD

;result: A = 99H
```

UTL_CONVERT_2BYTE_HEX_TO_2BYTE_BCD

Description Converts a 2-byte HEX number to 2-byte BCD number.

Usage UTL_CONVERT_2BYTE_HEX_TO_2BYTE_BCD

Assumptions None

Input HL - HEX data to be converted

Output BA - BCD equivalent

Destroys None

Example

```
;convert hex to BCD
ld  HL, #1234H
UTL_CONVERT_2BYTE_HEX_TO_2BYTE_BCD

;result: BA = 4460H
```

UTL_CONVERT_1BYTE_BCD_TO_1BYTE_HEX

Description Converts a 1-byte BCD number to 1-byte HEX number.

Usage UTL_CONVERT_1BYTE_BCD_TO_1BYTE_HEX

Assumptions None

Input A - BCD data to be converted

Output A - hexadecimal byte

Destroys B, HL

Example

```
;convert BCD to HEX
ld  A, #99H
UTL_CONVERT_1BYTE_BCD_TO_1BYTE_HEX

;result A = 63H
```

UTL_CONVERT_4BYTE_FROM_BCD_TO_HEX

Description Converts a 4 byte buffer from BCD to HEX.

Usage UTL_CONVERT_4BYTE_FROM_BCD_TO_HEX

Assumptions Dataset is stored in a consecutive 4 byte buffer, order least significant to most significant.

Input IX - Points to the least significant byte of the data buffer.

Output IX - Points to the most significant byte of the data buffer. The buffer contains the data in HEX format.

Destroys B

Example

```
MyBCDNumber:

   db  99H
   db  35H
   db  15H
   db  09H

;convert BCD to HEX
ld  IX, #MyBCDNumber
UTL_CONVERT_4BYTE_FROM_BCD_TO_HEX

;results stored in MyBCDNumber
;  MyBCDNumber:
;    db  63H
;    db  23H
;    db  0FH
;    db  09H
```

UTL_CONVERT_4BYTE_FROM_HEX_TO_BCD

Description Converts a 4 byte buffer from HEX to BCD.

Usage UTL_CONVERT_4BYTE_FROM_HEX_TO_BCD

Assumptions Dataset is stored in a consecutive 4 byte buffer, order least significant to most significant.

Input
IX - Points to the least significant byte of the data buffer.

Output
IX - Points to the most significant byte of the data buffer. The buffer contains the data in BCDformat.

Destroys B

Example

```
MyBCDNumber:
    db  63H
    db  23H
    db  0FH
    db  09H

;convert HEX to BCD
ld  IX, #MyBCDNumber
UTL_CONVERT_4BYTE_FROM_HEX_TO_BCD

;results stored in MyBCDNumber
;MyBCDNumber:
;
;   db  99H
;   db  35H
;   db  15H
;   db  09H
```

UTL_CONVERT_TO_12HR_FORMAT

Description Converts the 24 hour data into 12 hour data and determine if it is AM or PM.

Usage UTL_CONVERT_TO_12HR_FORMAT

Assumptions Data is in BCD format.

Input
A - Hour data to be formatted.

Output
A - 12 hour format data.
CarryFlag - TRUE if AM
FALSE if PM

Destroys None

Example

```
;convert the hour into 12hour format
ld  A, #22H
UTL_CONVERT_TO_12HR_FORMAT
;result A = 10H

jr  C,  DisplayAMText
jr  NC, DisplayPMText
```

Math Functions

UTL_DIVIDE_HMSH_BY_N_IN_HEX

Description Divides a data buffer of the format Hr:Min:Sec.Hun by a divisor N and places the result in the buffer.

Usage UTL_DIVIDE_HMSH_BY_N_IN_HEX

Assumptions Dataset is stored in a consecutive 4 byte buffer, order least significant to most significant.
Data is unsigned, in Hexadecimal format

Input

IX	-	Points to the most significant byte of the data buffer. (HEX data)
A		Divisor. (HEX data)

Output

IX	-	Points to the least significant byte of the data buffer. The buffer contains the data (result) in hexadecimal format.

Destroys HL

Example

```
MyTimeBuffer:

   ;Time: 01:30:16.50
   db   50H   ;hundredths
   db   16H   ;second
   db   30H   ;minute
   db   01H   ;hour

NumberOfLaps:
   db   0AH   ;10 laps
…

;convert time buffer from BCD to HEX
ld  IX, #MyBCDNumber
UTL_CONVERT_4BYTE_FROM_BCD_TO_HEX

ld  IX, #(MyTimeBuffer+3)   ;point to hour data (most significant byte)
ld  A, [NumberOfLaps]
UTL_DIVIDE_HMSH_BY_N_IN_HEX

;convert back result to BCD
ld  IX, #MyBCDNumber
UTL_CONVERT_4BYTE_FROM_HEX_TO_BCD

;result:
; Time: 00:09.01.65
;   db   65H   ;hundredths
```

```
;   db   01H   ;second
;   db   09H   ;minute
;   db   00H   ;hour
```

UTL_DIVIDE_HMSH_BY_N_IN_BCD

Description Divides a data buffer of the format Hr:Min:Sec.Hun by a divisor N and places the result in the buffer.

Usage UTL_DIVIDE_HMSH_BY_N_IN_BCD

Assumptions Dataset is stored in a consecutive 4 byte buffer, order least significant to most significant.
Data is unsigned, in Hexadecimal format

Input

IX	-	Points to the LSB of the data buffer (BCD Format).
A		Divisor. (HEX data)

Output

IX	-	Points to the most significant byte of the data buffer.

Destroys B, HL

Example

```
MyTimeBuffer:

   ;Time: 01:30:16.50
   db   50H   ;hundredths
   db   16H   ;second
   db   30H   ;minute
   db   01H   ;hour

NumberOfLaps:
   db   0AH   ;10 laps
…

ld  IX, #(MyTimeBuffer+3)   ;point to hour data (most significant byte)
ld  A, [NumberOfLaps]
UTL_DIVIDE_HMSH_BY_N_IN_BCD

;result:
; Time: 00:09.01.65
;   db   65H   ;hundredths
;   db   01H   ;second
;   db   09H   ;minute
;   db   00H   ;hour
```

UTL_ADD_HMSH

Description Adds data in the Util Hr:Min:Sec:Hun buffer to user data.

Usage UTL_ADD_HMSH

Assumptions Data is already loaded into the UTL Hr:Min:Sec:Hun (KRESxxxBuffer). The structure of the KRESxxxBuffer is shown below:

```
KRESHundBuffer          equ (KRESBuffer+0)
KRESSecBuffer           equ (KRESBuffer+1)
KRESMinBuffer           equ (KRESBuffer+2)
KRESHrBuffer            equ (KRESBuffer+3)
KRESHrRollOverValue     equ (KRESBuffer+4)
KRESHourRolloverExcess  equ (KRESBuffer+5)
```

Input

HL	-	Points to the hundredths data of the user buffer.

Output

HL	-	Points to last data location in buffer. (Hour data)
A	-	Set to indicate data that was modified after addition.

The register A will be have the following bits set:

Bit	Description
0	Seconds Changed
1	Minutes Changed
2	Hours Changed

Destroys May destroy the contents of the following variables: KRESHrBuffer, KRESMinBufferm KRESSecBuffer and KRESHundBuffer.

Example

```
MyUserData:

    ;user time: 01:05.10.50
    db  50H    ;hundredths
    db  10H    ;seconds
    db  05H    ;minute
    db  01H    ;hour
…

;clear all entries in the KRESBuffer
ld  B, #6
ld  HL, #KRESBuffer
KRES_CLEAR_N_BYTES

;setup the data to add and store in KRESBuffer
;add 30 minutes to the user data
ld  A, #30H
ld  [KRESMinBuffer], A

;add 30 minutes to user data
ld  HL, #MyUserData
UTL_ADD_HMSH

;result: new user time = 01:35.10.50
```

UTL_SUBTRACT_HMSH

Description Subtracts data in the UTL Hr:Min:Sec:Hun buffer from user data.

Usage UTL_SUBTRACT_HMSH

Assumptions Data is already loaded into the UTL Hr:Min:Sec:Hun (KRESxxxBuffer). The structure of the KRESxxxBuffer is shown below:

```
KRESHundBuffer           equ (KRESBuffer+0)
KRESSecBuffer            equ (KRESBuffer+1)
KRESMinBuffer            equ (KRESBuffer+2)
KRESHrBuffer             equ (KRESBuffer+3)
KRESHrRollOverValue      equ (KRESBuffer+4)
KRESHourRolloverExcess   equ (KRESBuffer+5)
```

Input

HL	-	Points to the hundredths data of the user buffer.

Output

HL	-	Points to last data location in buffer. (Hour data)
A	-	Set to indicate data that was modified after addition.

The register A will be have the following bits set:

Bit	Description
0	Seconds Changed
1	Minutes Changed
2	Hours Changed

Destroys May destroy the contents of the following variables: KRESHrBuffer, KRESMinBufferm KRESSecBuffer and KRESHundBuffer.

Example

```
MyUserData:

    ;user time: 01:05.10.50
    db  50H    ;hundredths
    db  10H    ;seconds
    db  05H    ;minute
    db  01H    ;hour
…

;clear all entries in the KRESBuffer
ld  B, #6
ld  HL, #KRESBuffer
KRES_CLEAR_N_BYTES

;setup the data to add and store in KRESBuffer
;subtract 30 minutes to the user data
ld  A, #30H
ld  [KRESMinBuffer], A

;subtract 30 minutes to user data
ld  HL, #MyUserData
UTL_SUBTRACT_HMSH

;result: new user time = 00:35.10.50
```

Copy

UTL_COPY_BUFFER

Description Copies the data pointed by IYReg to the location pointed by HLReg.

Usage UTL_COPY_BUFFER

Assumptions None

Input

B	-	Number of bytes to copy
IY	-	Base address of the source data
HL	-	Base address of the destination

Output None

Destroys None

Example

```
MyUserData:

    ;user time: 01:05.10.50
    db  50H    ;hundredths
    db  10H    ;seconds
    db  05H    ;minute
    db  01H    ;hour
…

;copy user data into KRESBuffer
ld  B,  #4
ld  HL, #KRESBuffer    ;destination buffer
ld  IY, #MyUserData    ;source buffer
UTL_COPY_BUFFER
```

UTL_COPY_IYREG_TO_IXREG

Description Copies the data pointed by IYReg to the location pointed by IXReg.

Usage UTL_COPY_IYREG_TO_IXREG

Assumptions None

Input

B	-	Number of bytes to copy
IY	-	Base address of the source data
IX	-	Base address of the destination

Output None

Destroys None

Example

```
MyUserData:
```

```
    db  50H    ;hundredths
    db  10H    ;seconds
    db  05H    ;minute
    db  01H    ;hour


MyWorkData:

    db  00H
    db  00H
    db  00H
    db  00H

…

;copy 4 bytes from MyUserData structure to the MyWorkData structure
ld  B, #4
ld  IY, #MyUserData
ld  IX, #MyWorkData
UTL_COPY_IYREG_TO_IXREG
```

Comparison

UTL_COMPARE_4BYTE_BUFFER

Description Compares the data in a 4 byte buffer, and sets the MCU condition flags to indicate if one data set is greater than, less than or equal.

Usage UTL_COMPARE_4BYTE_BUFFER

Assumptions None

Input

HL	-	points to most significant byte of data to be compared to
IX	-	points to most significany byte of data we are comparing

Output

`bZeroFlag = TRUE , bCarryFlag = FALSE`	-	HLReg = IXReg
`bZeroFlag = FALSE , bCarryFlag = TRUE`	-	HLReg < IXReg
`bZeroFlag = FALSE , bCarryFlag = FALSE`	-	HLReg > IXReg

Destroys BA, HL, IX

Example

```
MyUserData:

    db  50H    ;hundredths
    db  10H    ;seconds
    db  05H    ;minute
    db  01H    ;hour


MyWorkData:

    db  00H
    db  00H
    db  00H
```

```
    db  00H

…

;compare MyUserData with MyWorkData structures
ld  B, #4
ld  HL, #(MyUserData+3)   ;point to hour data (most significant byte)
ld  IX, #(MyWorkData+3)   ;point to hour data (most significant byte)
UTL_COMPARE_4BYTE_BUFFER

jr  Z,  TwoStructuresAreEqual
jr  C,  UserDataIsLessThanWorkData
jr  NC, UserDataIsGreaterThanWorkData
```

UTL_COMPARE_HLREG_WITH_IXREG

Description Compares two structures as pointed by HLReg and IXReg.

Usage UTL_COMPARE_HLREG_WITH_IXREG

Assumptions None

Input

HL	-	points to most significant byte of data to be compared to
IX	-	points to most significany byte of data we are comparing
B	-	Number of byte fields are to be compared

Output

`bZeroFlag = TRUE , bCarryFlag = FALSE`	-	HLReg = IXReg
`bZeroFlag = FALSE , bCarryFlag = TRUE`	-	HLReg < IXReg
`bZeroFlag = FALSE , bCarryFlag = FALSE`	-	HLReg > IXReg

Destroys BA, HL, IX

Example

```
MyUserData:

    db  50H
    db  10H
    db  05H
    db  01H
    db  01H
    db  01H
    db  01H

MyWorkData:

    db  00H
    db  00H
    db  00H
    db  00H
    db  00H
    db  00H
    db  00H

…

;compare MyUserData with MyWorkData structures
```

```
ld  B,  #7
ld  HL, #(MyUserData+6)   ;point to most significant byte
ld  IX, #(MyWorkData+6)   ;point to most significant byte
UTL_COMPARE_HLREG_WITH_IXREG

jr  Z,  TwoStructuresAreEqual
jr  C,  UserDataIsLessThanWorkData
jr  NC, UserDataIsGreaterThanWorkData
```

UTL_COMPARE_HLREG_WITH_IYREG

Description Compares two structures as pointed by HLReg and IYReg.

Usage UTL_COMPARE_HLREG_WITH_IXREG

Assumptions None

Input

HL	-	points to most significant byte of data to be compared to
IY	-	points to most significany byte of data we are comparing
B	-	Number of byte fields are to be compared

Output

`bZeroFlag = TRUE , bCarryFlag = FALSE`	-	HLReg = IXReg
`bZeroFlag = FALSE , bCarryFlag = TRUE`	-	HLReg < IXReg
`bZeroFlag = FALSE , bCarryFlag = FALSE`	-	HLReg > IXReg

Destroys BA, HL, IY

Example

```
MyUserData:

    db  50H
    db  10H
    db  05H
    db  01H
    db  01H
    db  01H
    db  01H

MyWorkData:

    db  00H
    db  00H
    db  00H
    db  00H
    db  00H
    db  00H
    db  00H

…

;compare MyUserData with MyWorkData structures
ld  B,  #7
ld  HL, #(MyUserData+6)   ;point to most significant byte
ld  IY, #(MyWorkData+6)   ;point to most significant byte
UTL_COMPARE_HLREG_WITH_IYREG
```

```
jr  Z,  TwoStructuresAreEqual
jr  C,  UserDataIsLessThanWorkData
jr  NC, UserDataIsGreaterThanWorkData
```

Acceleration

UTLACCELERATION_1SEC

Description 1 Sec acceleration table.

Usage UTLACCELERATION_1SEC

Assumptions None

Input COREEventArgument - Index into the acceleration table contains the number of pulses detected within 16hz window.

Output KRESSecBuffer the Acceleration value of data

Destroys HL, IY, IX

Example

```
;clear all entries in the KRESBuffer
ld  B, #6
ld  HL, #KRESBuffer
KRES_CLEAR_N_BYTES

;data to convert is already loaded in the COREEVENTArgument variable
UTLACCELERATION_1SEC

;result in KRESSecBuffer
```

UTL_ACCELERATION_1INCREMENT

Description Generic 1 unit acceleration table.

Usage UTL_ACCELERATION_1INCREMENT

Assumptions None

Input COREEventArgument - Index into the acceleration table contains the number of pulses detected within 16hz window.

Output KRESSecBuffer the Acceleration value of data

Destroys HL, IY, IX

Example

```
;clear all entries in the KRESBuffer
ld  B, #6
ld  HL, #KRESBuffer
KRES_CLEAR_N_BYTES

;data to convert is already loaded in the COREEVENTArgument variable
UTLACCELERATION_1INCREMENT

;result in KRESSecBuffer
```

UTLACCELERATION_1MIN

Description 1 Min acceleration table.

Usage UTLACCELERATION_1MIN

Assumptions Clears the KRESxxxBuffer first before loading the acceleration value.

Input	COREEventArgument	-	Index into the acceleration table contains the number of pulses detected within 16hz window.
Output	KRESMinBuffer		the Acceleration value of data

Destroys HL, IY, IX

Example

```
;clear all entries in the KRESBuffer
ld  B, #6
ld  HL, #KRESBuffer
KRES_CLEAR_N_BYTES

;data to convert is already loaded in the COREEVENTArgument variable
UTLACCELERATION_1MIN

;result in KRESMinBuffer
```

UTLACCELERATION_5MIN

Description 5 Min acceleration table

Usage UTLACCELERATION_5MIN

Assumptions Clears the KRESxxxBuffer first before loading the acceleration value.

Input	COREEventArgument	-	Index into the acceleration table contains the number of pulses detected within 16hz window.
Output	KRESMinBuffer		the Acceleration value of data

Destroys HL, IY, IX

Example
```
;clear all entries in the KRESBuffer
ld  B, #6
ld  HL, #KRESBuffer
KRES_CLEAR_N_BYTES

;data to convert is already loaded in the COREEVENTArgument variable
UTLACCELERATION_5MIN

;result in KRESMinBuffer
```

UTL_ACCELERATION_5INCREMENT

Description Generic 5 units accleration

Usage UTL_ACCELERATION_5INCREMENT

Assumptions Clears the KRESxxxBuffer first before loading the acceleration value.

Input COREEventArgument - Index into the acceleration table contains the number of pulses detected within 16hz window.

Output KRESMinBuffer the Acceleration value of data

Destroys HL, IY, IX

Example
```
;clear all entries in the KRESBuffer
ld  B, #6
ld  HL, #KRESBuffer
KRES_CLEAR_N_BYTES

;data to convert is already loaded in the COREEVENTArgument variable
UTLACCELERATION_5INCREMENT

;result in KRESMinBuffer
```

UTLACCELERATION_DATE

Description Get number of days from the utlAccelerationTable1Min acceleration table and return the number in A.

Usage UTLACCELERATION_DATE

Assumptions Clears the KRESxxxBuffer first before loading the acceleration value.

Input COREEventArgument - Index into the acceleration table contains the number of pulses detected within 16hz window.

Output A Number of days from the acceleration table for the given number of pulses.

Destroys None

Example

UTLACCELERATION_DATE

Description Get number of days from the utlAccelerationTable1Min acceleration table and return the number in A.

Usage UTLACCELERATION_DATE

Assumptions Clears the KRESxxxBuffer first before loading the acceleration value.

Input COREEventArgument - Index into the acceleration table contains the number of pulses detected within 16hz window.

Output A Number of days from the acceleration table for the given number of pulses.

Destroys None

Example

```
;clear all entries in the KRESBuffer
ld  B, #6
ld  HL, #KRESBuffer
KRES_CLEAR_N_BYTES

;data to convert is already loaded in the COREEVENTArgument variable
UTLACCELERATION_DATE

;result in A register
```

Message Editing

UTL_SETUP_VARS_FOR_EDITING

Description Sets up the variables used for editing a message. This includes the sentinel character.

Usage UTL_SETUP_VARS_FOR_EDITING

Assumptions LCDScrollBuffer should contain the message to be edited.

Input B - Number of characters allocated for the message.

Output None

Destroys None

Example
```
LCDScrollBuffer:
    db   DM5_A, DM5_B DM5_C, DM5_D, DM5_E, DM5_F, DM5_G, DM5_H
    db   DM5_I, DM5_J, DM5_K, DM5_L, DM5_M, DM5_N, DM5_O, DM5_P
    db   DM5_SENTINEL
…

ld  B, #17
UTL_SETUP_VARS_FOR_EDITING
```

UTL_FILL_SCROLL_BUFFER_WITH_SPACE

Description Fills up the scroll buffer with space character.

Usage UTL_FILL_SCROLL_BUFFER_WITH_SPACE

Assumptions LCDScrollBuffer should contain the message to be edited.

Input None

Output None

Destroys None

Example
```
LCDScrollBuffer:
    db   DM5_A, DM5_B DM5_C, DM5_D, DM5_E, DM5_F, DM5_G, DM5_H
    db   DM5_I, DM5_J, DM5_K, DM5_L, DM5_M, DM5_N, DM5_O, DM5_P
    db   DM5_SENTINEL
…

;fill up scroll buffer (101 bytes total) with space
UTL_FILL_SCROLL_BUFFER_WITH_SPACE

;RESULT:  all 101 bytes in LCDScrollBuffer will be set to DM5_SPACE
```

UTL_MOVE_CURSOR_FORWARD_AND_REQ_BLINK

Description Move to the next character to be edited if it is not yet the last character.
Display cursor on the position of the character to be edited.
Request blinking on the character to be edited.

Usage UTL_MOVE_CURSOR_FORWARD_AND_REQ_BLINK

Assumptions LCDScrollBuffer should contain the message to be edited.

Input None

Output bCarryFlag - 0 -- if moving the cursor forward is valid.
1 -- if moving the cursor forward is invalid

Destroys BA, IX, IY, HL

Example

```
LCDScrollBuffer:
    db   DM5_A, DM5_B DM5_C, DM5_D, DM5_E, DM5_F, DM5_G, DM5_H
    db   DM5_I, DM5_J, DM5_K, DM5_L, DM5_M, DM5_N, DM5_O, DM5_P
    db   DM5_SENTINEL
…

;setup for message editing
ld  B, #17
UTL_SETUP_VARS_FOR_EDITING

;display the section of message where character is to be edited
UTL_DISPLAY_MSG_AND_REQ_BLINK

;move cursor forward
UTL_MOVE_CURSOR_FORWARD_AND_REQ_BLINK
```

UTL_MOVE_CURSOR_BACKWARD_AND_REQ_BLINK

Description Move to the previous character to be edited if its not yet the first character.
Display cursor on the position of the character to be edited.
Request blinking on the character to be edited.

Usage UTL_MOVE_CURSOR_BACKWARD_AND_REQ_BLINK

Assumptions LCDScrollBuffer should contain the message to be edited.

Input None

Output bCarryFlag - 0 -- if moving the cursor backward is valid.
1 -- if moving the cursor backward is invalid

Destroys BA, IX, IY, HL

Example

```
LCDScrollBuffer:
    db   DM5_A, DM5_B DM5_C, DM5_D, DM5_E, DM5_F, DM5_G, DM5_H
    db   DM5_I, DM5_J, DM5_K, DM5_L, DM5_M, DM5_N, DM5_O, DM5_P
    db   DM5_SENTINEL
…

;setup for message editing
ld  B, #17
UTL_SETUP_VARS_FOR_EDITING

;display the section of message where character is to be edited
UTL_DISPLAY_MSG_AND_REQ_BLINK

;move cursor forward
UTL_MOVE_CURSOR_FORWARD_AND_REQ_BLINK
```

```
;move cursor forward
UTL_MOVE_CURSOR_FORWARD_AND_REQ_BLINK

;move cursor forward
UTL_MOVE_CURSOR_FORWARD_AND_REQ_BLINK

;move cursor backward
UTL_MOVE_CURSOR_BACKWARD_AND_REQ_BLINK

;move cursor backward
UTL_MOVE_CURSOR_BACKWARD_AND_REQ_BLINK
```

UTL_DISPLAY_MSG_AND_REQ_BLINK

Description Displays a part of the message that has the character to be edited.
The cursor character will be displayed on the position of the character to be edited.

Usage UTL_DISPLAY_MSG_AND_REQ_BLINK

Assumptions LCDScrollBuffer should contain the message to be edited.

Input None

Output None

Destroys BA, IX, IY, HL

Example

```
LCDScrollBuffer:
    db   DM5_A, DM5_B DM5_C, DM5_D, DM5_E, DM5_F, DM5_G, DM5_H
    db   DM5_I, DM5_J, DM5_K, DM5_L, DM5_M, DM5_N, DM5_O, DM5_P
    db   DM5_SENTINEL
…

;display the section of message where character is to be edited
UTL_DISPLAY_MSG_AND_REQ_BLINK
```

UTL_POINT_TO_PREV_CHAR_AND_REQ_BLINK

Description Point to the previous character (refer to lcd character table on whats the previous character with respect to the character to be changed).
Displays the new character and request blinking.

Usage UTL_POINT_TO_PREV_CHAR_AND_REQ_BLINK

Assumptions LCDScrollBuffer should contain the message to be edited.

Input B - Number to be subtracted to the current character definition to point to the next

desired character. This would look like it is accelerated

Output None

Destroys BA, IX, IY, HL

Example

```
LCDScrollBuffer:
    db   DM5_A, DM5_B DM5_C, DM5_D, DM5_E, DM5_F, DM5_G, DM5_H
    db   DM5_I, DM5_J, DM5_K, DM5_L, DM5_M, DM5_N, DM5_O, DM5_P
    db   DM5_SENTINEL
…

;setup for message editing
ld  B, #17
UTL_SETUP_VARS_FOR_EDITING

;display the section of message where character is to be edited
UTL_DISPLAY_MSG_AND_REQ_BLINK
…

;change character under cursor to the next available character index
;with wraparound with the offset position defined in the
;COREEventArgument variable
ld  B,[COREEventArgument]
UTL_POINT_TO_PREV_CHAR_AND_REQ_BLINK
```

UTL_POINT_TO_NEXT_CHAR_AND_REQ_BLINK

Description Point to the next character (refer to lcd character table on what is the next character with respect to the character to be changed).
Displays the new character and request blinking.

Usage UTL_POINT_TO_NEXT_CHAR_AND_REQ_BLINK

Assumptions LCDScrollBuffer should contain the message to be edited.

Input B - Number to be subtracted to the current character definition to point to the next desired character. This would look like it is accelerated

Output None

Destroys BA, IX, IY, HL

Example

```
LCDScrollBuffer:
    db   DM5_A, DM5_B DM5_C, DM5_D, DM5_E, DM5_F, DM5_G, DM5_H
    db   DM5_I, DM5_J, DM5_K, DM5_L, DM5_M, DM5_N, DM5_O, DM5_P
    db   DM5_SENTINEL
…

;setup for message editing
ld  B, #17
UTL_SETUP_VARS_FOR_EDITING

;display the section of message where character is to be edited
```

```
UTL_DISPLAY_MSG_AND_REQ_BLINK
…

;change character under cursor to the previous available character index
;with wraparound with the offset position defined in the
;COREEventArgument variable
ld  B,[COREEventArgument]
UTL_POINT_TO_PREV_CHAR_AND_REQ_BLINK
```

UTL_POINT_TO_PREV_CHAR

Description Point to the previous character (refer to lcd character table on what is the previous character with respect to the character to be changed).

Usage UTL_POINT_TO_PREV_CHAR

Assumptions None

Input

IY	Base address of the message where the character will be edited
L	Offset from the base address of the character to be edited
B	Number to be subtracted to the current character definition to point to the next desired character. This would look like it is accelerated.

Output

A	New character. The new character is stored on the position pointed by (IYReg + LReg).

Destroys None

Example

```
MyCityCode:
    db   DM5_N, DM5_Y DM5_C
…

;change character specified to the previous available character index
;with wraparound with the offset position defined in the
;COREEventArgument variable
ld  IY, #MyCityCode
ld  L,  #1                  ;edit the 2ND city code
ld  B,  #1                  ;get previous character in set
UTL_POINT_TO_PREV_CHAR

;result
;MyCityCode:
;    db   DM5_N, DM5_X, DM5_C
```

UTL_POINT_TO_NEXT_CHAR

Description Point to the next character (refer to lcd character table on what is the previous character with respect to the character to be changed).

Usage UTL_POINT_TO_NEXT_CHAR

Assumptions None

Input

IY	Base address of the message where the character will be edited
L	Offset from the base address of the character to be edited
B	Number to be subtracted to the current charater definition to point to the next desired character. This would look like it is accelerated.

Output

A	New character. The new character is stored on the position pointed by (IYReg + LReg).

Destroys None

Example

```
MyCityCode:
    db   DM5_N, DM5_Y DM5_C
…

;change character specified to the next available character index
;with wraparound with the offset position defined in the
;COREEventArgument variable
ld  IY, #MyCityCode
ld  L,  #1                  ;edit the 2ND city code
ld  B,  #1                  ;get next character in set
UTL_POINT_TO_NEXT_CHAR

;result
;MyCityCode:
;    db   DM5_N, DM5_Z, DM5_C
```

UTL_CLEANUP_EDIT_BUFFER

Description Trims the trailing spaces.

Usage UTL_CLEANUP_EDIT_BUFFER

Assumptions LCDScrollBuffer should contain the message to be edited.

Input None

Output None

Destroys None

Example

```
LCDScrollBuffer:
    ;message “AB DEFG IJ   “ + SENTINEL
    db   DM5_A, DM5_B DM5_SPACE, DM5_D, DM5_E, DM5_F, DM5_G, DM5_SPACE
    db   DM5_I, DM5_J, DM5_ SPACE, DM5_ SPACE, DM5_ SPACE, DM5_ SPACE,
    db   DM5_SENTINEL
…

;clean up edit buffer of trailing spaces
```

```
UTL_CLEANUP_EDIT_BUFFER

;result
;MyCityCode:
;    ;message “AB DEFG IJ“ + SENTINEL
;    db   DM5_A, DM5_B DM5_SPACE, DM5_D, DM5_E, DM5_F, DM5_G, DM5_SPACE
;    db   DM5_I, DM5_J, DM5_SENTINEL, DM5_SENTINEL, DM5_SENTINEL
;    db   DM5_SENTINEL, DM5_SENTINEL, DM5_SENTINEL
;    db   DM5_SENTINEL
```

UTL_TRIM_SENTINEL_CHAR

Description Trims the trailing sentinel character. The last sentinel character is not included.

Usage UTL_TRIM_SENTINEL_CHAR

Assumptions LCDScrollBuffer should contain the message to be edited.

Input None

Output None

Destroys None

Example

```
LCDScrollBuffer:
    ;message “AB DEFG IJ“ + SENTINEL
    db   DM5_A, DM5_B DM5_SPACE, DM5_D, DM5_E, DM5_F, DM5_G, DM5_SPACE
    db   DM5_I, DM5_J, DM5_SENTINEL, DM5_SENTINEL, DM5_SENTINEL
    db   DM5_SENTINEL, DM5_SENTINEL, DM5_SENTINEL
    db   DM5_SENTINEL
…

;setup for message editing
ld  B, #17
UTL_SETUP_VARS_FOR_EDITING

;setup the buffer with spaces until the end of the valid edit buffer size
UTL_TRIM_SENTINEL_CHAR

;result
;MyCityCode:
;    ;message “AB DEFG IJ“ + SENTINEL
;    db   DM5_A, DM5_B DM5_SPACE, DM5_D, DM5_E, DM5_F, DM5_G, DM5_SPACE
;    db   DM5_I, DM5_J, DM5_SPACE, DM5_SPACE, DM5_SPACE, DM5_SPACE,
;    db   DM5_SPACE, DM5_SPACE
;    db   DM5_SENTINEL
```

UTL_CHANGE_CHAR_TO_SPACE

Description Change the current character into a space character.

Usage UTL_CHANGE_CHAR_TO_SPACE

Assumptions LCDScrollBuffer should contain the message to be edited.

Input None

Output None

Destroys None

Example

```
LCDScrollBuffer:
    db   DM5_A, DM5_B DM5_C, DM5_D, DM5_E, DM5_F, DM5_G, DM5_H
    db   DM5_I, DM5_J, DM5_K, DM5_L, DM5_M, DM5_N, DM5_O, DM5_P
    db   DM5_SENTINEL
…

;setup for message editing
ld  B, #17
UTL_SETUP_VARS_FOR_EDITING

;display the section of message where character is to be edited
UTL_DISPLAY_MSG_AND_REQ_BLINK

;move cursor forward
UTL_MOVE_CURSOR_FORWARD_AND_REQ_BLINK

;change character under cursor to a space
UTL_CHANGE_CHAR_TO_SPACE

; result
;LCDScrollBuffer:
;   db   DM5_A, DM5_SPACE, DM5_C, DM5_D, DM5_E, DM5_F, DM5_G, DM5_H
;   db   DM5_I, DM5_J, DM5_K, DM5_L, DM5_M, DM5_N, DM5_O, DM5_P
;   db   DM5_SENTINEL
```

Time Structure Math

The following API uses the KRESBuffer as the default buffer where data to be added or subtracted from the user data structure of the same format.

Offset	Variable Name
0	KRESHundBuffer
1	KRESSecBuffer
2	KRESMinBuffer
3	KRESHrBuffer
4	KRESHrRollOverValue
5	KRESHourRolloverExcess

The following are the bit definitions that indicate what data has been updated:

Bit Position	Variable Name
00000001B	bKRESSecUpd
00000010B	bKRESMinUpd
00000100B	bKRESHrUpd
00001000B	bKRESDayUpd

KRES_CLEARDATABUFFERS

Description Clears the KRESHundBuffer,KRESSecBuffer, KRESMinBuffer,KRESHrBuffer and KRESHrRollOverValue.

Usage KRES_CLEARDATABUFFERS

Assumptions None

Input None

Output None

Destroys HL

Example

```
;clear all variables in the buffer prior to loading the required
;data for addition
KRES_CLEARDATABUFFERS

;load new hour and minute
ld  A, [NumberOfHoursToAdd]
ld  [KRESHrBuffer], A
ld  A, [NumberOfMinutesToAdd]
ld  [KRESMinBuffer], A

;setup hour rollover value
ld  A, #24H
ld  [KRESHrRollOverValue], A

;add hour and minute
ld   HL, #MyTimeStructure
KRES_MINHR_UPD

…

MyTimeStructure:

   MyMinute:    ds    1
   MyHour:      ds    1
```

KRES_HUNDSECMINHR_UPD

Description Updates the data stucture HH:MM:SS.hh by adding the buffer HH:MM:SS.hh to the data structure.

Usage KRES_HUNDSECMINHR_UPD

Assumptions KRESBuffer is already loaded with the required data to add into the user time structure.

Input
HL - Points to the hundredths data
KRESHrRollOverValue - Maximum hour rollover value

Output A update flags

Destroys May destroy the contents of the following variables: KRESHrBuffer, KRESMinBufferm KRESSecBuffer and KRESHundBuffer.

Example

```
;clear all variables in the buffer prior to loading the required
;data for addition
KRES_CLEARDATABUFFERS

;load number of hundredths to add
ld  A, [NumberOfHundredthsToAdd]
ld  [KRESHundBuffer], A

;setup hour rollover value
ld  A, #24H
ld  [KRESHrRollOverValue], A

;add in the accumulated hundredths into the structure
ld   HL, #MyHundred
KRES_HUNDSECMINHR_UPD

…

MyTimeStructure:

   MyHundred:   ds    1
   MySecond:    ds    1
   MyMinute:    ds    1
   MyHour:      ds    1
```

KRES_SECMINHR_UPD

Description Updates the HH:MM:SS structure by adding the buffer HH:MM:SS to the data structure.

Usage KRES_SECMINHR_UPD

Assumptions KRESBuffer is already loaded with the required data to add into the user time structure.

Input
HL - Points to the seconds data
KRESHrRollOverValue - Maximum hour rollover value

Output A update flags

Destroys May destroy the contents of the following variables: KRESHrBuffer, KRESMinBufferm

KRESSecBuffer and KRESHundBuffer.

Example

```
;clear all variables in the buffer prior to loading the required
;data for addition
KRES_CLEARDATABUFFERS

;load number of hundredths to add
ld  A, [NumberOfSecondsToAdd]
ld  [KRESSecBuffer], A

;setup hour rollover value
ld  A, #24H
ld  [KRESHrRollOverValue], A

;add in the accumulated seconds into the structure
ld   HL, #MySecond
KRES_SECMINHR_UPD

…

MyTimeStructure:

   MyHundred:   ds    1
   MySecond:    ds    1
   MyMinute:    ds    1
   MyHour:      ds    1
```

KRES_MINHR_UPD

Description Updates the HH:MM structure by adding the buffer HH:MM to the data structure.

Usage KRES_MINHR_UPD

Assumptions KRESBuffer is already loaded with the required data to add into the user time structure.

Input

HL	-	Points to the minutes data
KRESHrRollOverValue	-	Maximum hour rollover value

Output

A	-	update flags
KRESHourRolloverExcess	-	Number of days to subtract resulting from API execution

Destroys May destroy the contents of the following variables: KRESHrBuffer, KRESMinBufferm KRESSecBuffer and KRESHundBuffer.

Example

```
;clear all variables in the buffer prior to loading the required
;data for addition
KRES_CLEARDATABUFFERS

;load number of minutes and hours to add
ld  A, [NumberOfMinutesToAdd]
ld  [KRESMinBuffer], A
ld  A, [NumberOfHoursToAdd]
ld  [KRESHrBuffer], A

;setup hour rollover value
```

```
ld  A, #24H
ld  [KRESHrRollOverValue], A

;add in the accumulated minutes into the structure
ld   HL, #MyMinute
KRES_MINHR_UPD

…

MyTimeStructure:

   MyHundred:   ds    1
   MySecond:    ds    1
   MyMinute:    ds    1
   MyHour:      ds    1
```

KRES_HUNDSECMINHR_SUB

Description subtracts the HH:MM:SS.hh buffer from HH:MM:SS.hh data strcture and store the result at data structure.

Usage KRES_HUNDSECMINHR_SUB

Assumptions KRESBuffer is already loaded with the required data to add into the user time structure.

Input

HL	-	Points to the hundredths data
KRESHrRollOverValue	-	Maximum hour rollover value

Output A update flags

Destroys May destroy the contents of the following variables: KRESHrBuffer, KRESMinBufferm KRESSecBuffer and KRESHundBuffer.

Example

```
;clear all variables in the buffer prior to loading the required
;data for addition
KRES_CLEARDATABUFFERS

;load number of hundredths to subtract
ld  A, [NumberOfHundredths]
ld  [KRESHundBuffer], A

;setup hour rollover value
ld  A, #24H
ld  [KRESHrRollOverValue], A

;subtract the accumulated hundredths into the structure
ld   HL, #MyHundred
KRES_HUNDSECMINHR_SUB

…

MyTimeStructure:

   MyHundred:   ds    1
   MySecond:    ds    1
```

```
   MyMinute:    ds    1
   MyHour:      ds    1
```

KRES_SECMINHR_SUB

Description subtracts the buffer HH:MM:SS from HH:MM:SS structure
and store the result at data structure.

Usage KRES_SECMINHR_SUB

Assumptions KRESBuffer is already loaded with the required data to add into the user time structure.

Input

HL	-	Points to the seconds data
KRESHrRollOverValue	-	Maximum hour rollover value

Output A update flags

Destroys May destroy the contents of the following variables: KRESHrBuffer, KRESMinBufferm KRESSecBuffer and KRESHundBuffer.

Example

```
;clear all variables in the buffer prior to loading the required
;data for addition
KRES_CLEARDATABUFFERS

;load number of hundredths to subtract
ld  A, [NumberOfSeconds]
ld  [KRESSecBuffer], A

;setup hour rollover value
ld  A, #24H
ld  [KRESHrRollOverValue], A

;subtract the accumulated seconds into the structure
ld   HL, #MySecond
KRES_SECMINHR_SUB

…

MyTimeStructure:

   MyHundred:   ds    1
   MySecond:    ds    1
   MyMinute:    ds    1
   MyHour:      ds    1
```

KRES_MINHR_SUB

Description Subtracts the buffer HH:MM from HH:MM structure and store the result at data structure.

Usage KRES_MINHR_SUB

Assumptions KRESBuffer is already loaded with the required data to add into the user time structure.

Input

HL	-	Points to the minute data
KRESHrRollOverValue	-	Maximum hour rollover value

Output

A	update flags
KRESHourRolloverExcess	Number of days to subtract resulting from API execution

Destroys May destroy the contents of the following variables: KRESHrBuffer, KRESMinBufferm KRESSecBuffer and KRESHundBuffer.

Example

```
;clear all variables in the buffer prior to loading the required
;data for addition
KRES_CLEARDATABUFFERS

;load number of minutes and hours to subtract
ld  A, [NumberOfMinutes]
ld  [KRESMinBuffer], A
ld  A, [NumberOfHours]
ld  [KRESHrBuffer], A

;setup hour rollover value
ld  A, #24H
ld  [KRESHrRollOverValue], A

;subtract accumulated minute and hour data into the structure
ld   HL, #MyMinute
KRES_MINHR_SUB

…

MyTimeStructure:

   MyHundred:   ds    1
   MySecond:    ds    1
   MyMinute:    ds    1
   MyHour:      ds    1
```

Trademarks