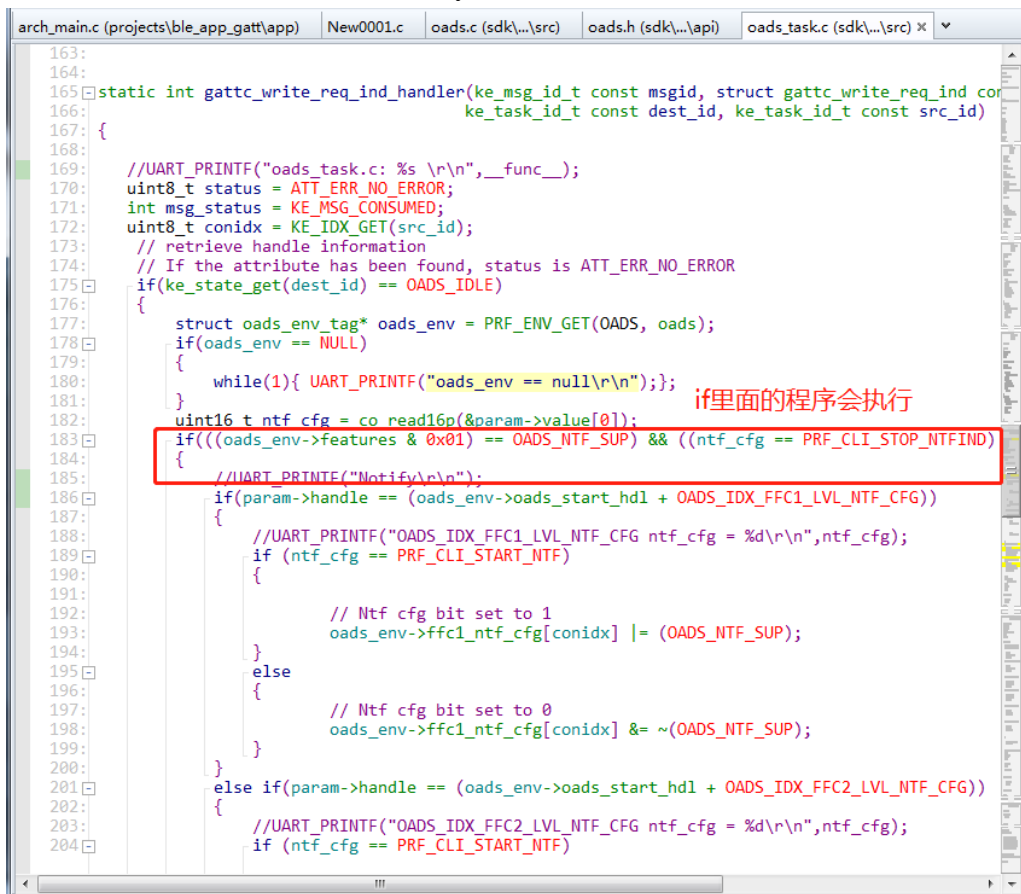


打开RC OTA，找到对应的蓝牙设备连接上后。

1. 点击OTA按钮，APP打开FFC1通道的notify监听，同时APP执行FFC1通道的write动作；蓝牙程序进入gattc\_write\_req\_ind\_handler函数。

(1) APP打开FFC1通道的notify监听，蓝牙端执行的对应程序：

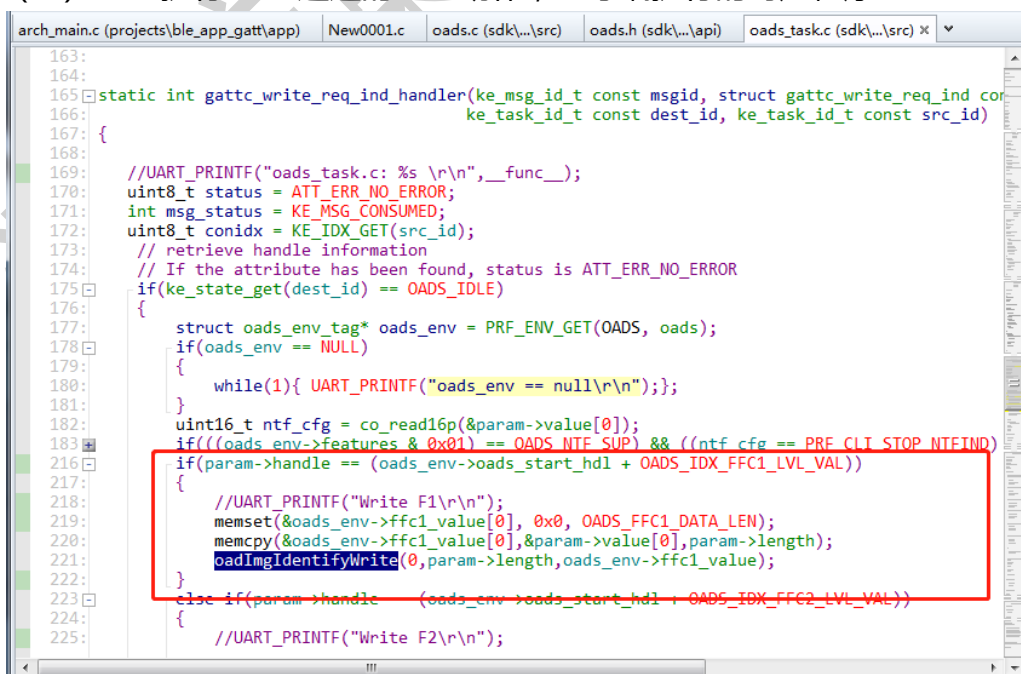


```

163:
164:
165: static int gattc_write_req_ind_handler(ke_msg_id_t const msgid, struct gattc_write_req_ind con
166:                                     ke_task_id_t const dest_id, ke_task_id_t const src_id)
167: {
168:
169:     //UART_PRINTF("oads_task.c: %s \r\n", __func__);
170:     uint8_t status = ATT_ERR_NO_ERROR;
171:     int msg_status = KE_MSG_CONSUMED;
172:     uint8_t conidx = KE_IDX_GET(src_id);
173:     // retrieve handle information
174:     // If the attribute has been found, status is ATT_ERR_NO_ERROR
175:     if(ke_state_get(dest_id) == OADS_IDLE)
176:     {
177:         struct oads_env_tag* oads_env = PRF_ENV_GET(OADS, oads);
178:         if(oads_env == NULL)
179:         {
180:             while(1){ UART_PRINTF("oads_env == null\r\n");};
181:
182:             uint16_t ntf_cfg = co_read16p(&param->value[0]);
183:             if(((oads_env->features & 0x01) == OADS_NTF_SUP) && ((ntf_cfg == PRF_CLI_STOP_NTFIND))
184:             {
185:                 //UART_PRINTF("Notify\r\n");
186:                 if(param->handle == (oads_env->oads_start_hdl + OADS_IDX_FFC1_LVL_NTF_CFG))
187:                 {
188:                     //UART_PRINTF("OADS_IDX_FFC1_LVL_NTF_CFG ntf_cfg = %d\r\n", ntf_cfg);
189:                     if (ntf_cfg == PRF_CLI_START_NTF)
190:                     {
191:
192:                         // Ntf cfg bit set to 1
193:                         oads_env->ffc1_ntf_cfg[conidx] |= (OADS_NTF_SUP);
194:                     }
195:                     else
196:                     {
197:                         // Ntf cfg bit set to 0
198:                         oads_env->ffc1_ntf_cfg[conidx] &= ~(OADS_NTF_SUP);
199:                     }
200:                 }
201:                 else if(param->handle == (oads_env->oads_start_hdl + OADS_IDX_FFC2_LVL_NTF_CFG))
202:                 {
203:                     //UART_PRINTF("OADS_IDX_FFC2_LVL_NTF_CFG ntf_cfg = %d\r\n", ntf_cfg);
204:                     if (ntf_cfg == PRF_CLI_START_NTF)

```

(2) APP执行FFC1通道的write动作，蓝牙端执行的对应程序：



```

163:
164:
165: static int gattc_write_req_ind_handler(ke_msg_id_t const msgid, struct gattc_write_req_ind con
166:                                     ke_task_id_t const dest_id, ke_task_id_t const src_id)
167: {
168:
169:     //UART_PRINTF("oads_task.c: %s \r\n", __func__);
170:     uint8_t status = ATT_ERR_NO_ERROR;
171:     int msg_status = KE_MSG_CONSUMED;
172:     uint8_t conidx = KE_IDX_GET(src_id);
173:     // retrieve handle information
174:     // If the attribute has been found, status is ATT_ERR_NO_ERROR
175:     if(ke_state_get(dest_id) == OADS_IDLE)
176:     {
177:         struct oads_env_tag* oads_env = PRF_ENV_GET(OADS, oads);
178:         if(oads_env == NULL)
179:         {
180:             while(1){ UART_PRINTF("oads_env == null\r\n");};
181:
182:             uint16_t ntf_cfg = co_read16p(&param->value[0]);
183:             if(((oads_env->features & 0x01) == OADS_NTF_SUP) && ((ntf_cfg == PRF_CLI_STOP_NTFIND))
216:             if(param->handle == (oads_env->oads_start_hdl + OADS_IDX_FFC1_LVL_VAL))
217:             {
218:                 //UART_PRINTF("Write F1\r\n");
219:                 memset(&oads_env->ffc1_value[0], 0x00, OADS_FFC1_DATA_LEN);
220:                 memcpy(&oads_env->ffc1_value[0], &param->value[0], param->length);
221:                 oad_img_identify_write(0, param->length, oads_env->ffc1_value);
222:             }
223:             else if(param->handle == (oads_env->oads_start_hdl + OADS_IDX_FFC2_LVL_VAL))
224:             {
225:                 //UART_PRINTF("Write F2\r\n");

```

- (3) 蓝牙端读出蓝牙设备程序的ver版本号和rom版本号（在0x20010地址上），蓝牙notify回传给APP。APP获取版本号后和升级的bin文件的版本号对比（如果ver相等，rom不相等，判断为部分升级；如果ver不相等，rom相等，判断为全部升级）

```

210: }
211:
212: void appm_update_param(struct gapc_conn_param *conn_param);
213: uint8_t oadImgIdentifyWrite(uint16_t connHandle, uint16_t length, uint8_t *pValue
214: {
215:     UART_PRINTF("oads.c:%s line:%d\r\n", __func__, __LINE__);
216:     img_hdr_t rxHdr;
217:     img_hdr_t ImgHdr;
218:     rxHdr.ver = co_read16p(&(pValue[4]));
219:     UART_PRINTF("rxHdr.ver. = %x \r\n", rxHdr.ver);
220:     rxHdr.len = co_read16p(&(pValue[6]));
221:     UART_PRINTF("rxHdr.len. = %x \r\n", rxHdr.len);
222:     rxHdr.uid = co_read32p(&(pValue[8]));
223:     UART_PRINTF("rxHdr.uid. = %x \r\n", rxHdr.uid);
224:     rxHdr.rom_ver = co_read16p(&(pValue[14]));
225:     UART_PRINTF("rxHdr.rom_ver. = %x \r\n", rxHdr.rom_ver);
226:     if(rxHdr.uid == OAD_APP_PART_UID) ...
227:     else if(rxHdr.uid == OAD_APP_STACK_UID) //全部OTA : bin文件的版本信息 ...
228:     {
229:         oad_firmware_type = 0;
230:         flash_read(0, SEC_IMAGE_OAD_HEADER_APP_FADDR, sizeof(img_hdr_t), (uint8_t *)&ImgHdr, NU
231:         UART_PRINTF("ImgHdr.rom_ver. = %x \r\n", ImgHdr.rom_ver);
232:         UART_PRINTF("ImgHdr.uid. = %x \r\n", ImgHdr.uid);
233:         UART_PRINTF("ImgHdr.ver. = %x \r\n", ImgHdr.ver);
234:     }
235:     if(oad_uid_check_status == 1) ...
236:     else
237:     {
238:         oadImgIdentifyReq(connHandle, &ImgHdr);
239:     }
240:     return ( 0x00 ); //SUCCESS
241: }
242:
243: /*****
244:  * @fn      oadImgBlockWrite
245:  * @brief   Process the Image Block Write.
246:  * @param   connHandle - connection message was received on
247:  * @param   pValue - pointer to data to be written
248:  */
249:
250:
251:
252:
253:
254:
255:
256:
257:
258:
259:
260:
261:
262:
263:
264:
265:
266:
267:
268:
269:
270:
271:
272:
273:
274:
275:
276:
277:
278:
279:
280:
281:
282:
283:
284:
285:
286:
287:
288:
289:
290:
291:
292:
293:
294:
295:
296:
297:
298:
299:
300:
301:
302:
303:
304:
305:
306:
307:
308:
309:
310:
311:
312:
313:
314:
315:
316:
317:
318:
319:
320:
321:
322:
323:
324:
325:
326:
327:
328:
329:
330:
331:
332:
333:
334:
335:
336:
337:
338:
339:
340:
341:
342:
343:
344:
345:
346:
347:
348:
349:
350:
351:
352:
353:
354:
355:
356:
357:
358:
359:
360:
361:
362:
363:
364:
365:
366:
367:
368:
369:
370:
371:
372:
373:
374:
375:
376:
377:
378:
379:
380:
381:
382:
383:
384:
385:
386:
387:
388:
389:
390:
391:
392:
393:
394:
395:
396:
397:
398:
399:
400:
401:
402:
403:
404:
405:
406:
407:
408:
409:
410:
411:
412:
413:
414:
415:
416:
417:
418:
419:
420:
421:
422:
423:
424:
425:
426:
427:
428:
429:
430:
431:
432:
433:
434:
435:
436:
437:
438:
439:
440:
441:
442:
443:
444:
445:
446:
447:
448:
449:
450:
451:
452:
453:
454:
455:
456:
457:
458:
459:
460:
461:
462:
463:
464:
465:
466:
467:
468:
469:
470:
471:
472:
473:
474:
475:
476:
477:
478:
479:
480:
481:
482:
483:
484:
485:
486:
487:
488:
489:
490:
491:
492:
493:
494:
495:
496:
497:
498:
499:
500:
501:
502:
503:
504:
505:
506:
507:
508:
509:
510:
511:
512:
513:
514:
515:
516:
517:
518:
519:
520:
521:
522:
523:
524:
525:
526:
527:
528:
529:
530:
531:
532:
533:
534:
535:
536:
537:
538:
539:
540:
541:
542:
543:
544:
545:
546:
547:
548:
549:
550:
551:
552:
553:
554:
555:
556:
557:
558:
559:
560:
561:
562:
563:
564:
565:
566:
567:
568:
569:
570:
571:
572:
573:
574:
575:
576:
577:
578:
579:
580:
581:
582:
583:
584:
585:
586:
587:
588:
589:
590:
591:
592:
593:
594:
595:
596:
597:
598:
599:
600:
601:
602:
603:
604:
605:
606:
607:
608:
609:
610:
611:
612:
613:
614:
615:
616:
617:
618:
619:
620:
621:
622:
623:
624:
625:
626:
627:
628:
629:
630:
631:
632:
633:
634:
635:
636:
637:
638:
639:
640:
641:
642:
643:
644:
645:
646:
647:
648:
649:
650:
651:
652:
653:
654:
655:
656:
657:
658:
659:
660:
661:
662:
663:
664:
665:
666:
667:
668:
669:
670:
671:
672:
673:
674:
675:
676:
677:
678:
679:
680:
681:
682:
683:
684:
685:
686:
687:
688:
689:
690:
691:
692:
693:
694:
695:
696:
697:
698:
699:
700:
701:
702:
703:
704:
705:
706:
707:
708:
709:
710:
711:
712:
713:
714:
715:
716:
717:
718:
719:
720:
721:
722:
723:
724:
725:
726:
727:
728:
729:
730:
731:
732:
733:
734:
735:
736:
737:
738:
739:
740:
741:
742:
743:
744:
745:
746:
747:
748:
749:
750:
751:
752:
753:
754:
755:
756:
757:
758:
759:
760:
761:
762:
763:
764:
765:
766:
767:
768:
769:
770:
771:
772:
773:
774:
775:
776:
777:
778:
779:
780:
781:
782:
783:
784:
785:
786:
787:
788:
789:
790:
791:
792:
793:
794:
795:
796:
797:
798:
799:
800:
801:
802:
803:
804:
805:
806:
807:
808:
809:
810:
811:
812:
813:
814:
815:
816:
817:
818:
819:
820:
821:
822:
823:
824:
825:
826:
827:
828:
829:
830:
831:
832:
833:
834:
835:
836:
837:
838:
839:
840:
841:
842:
843:
844:
845:
846:
847:
848:
849:
850:
851:
852:
853:
854:
855:
856:
857:
858:
859:
860:
861:
862:
863:
864:
865:
866:
867:
868:
869:
870:
871:
872:
873:
874:
875:
876:
877:
878:
879:
880:
881:
882:
883:
884:
885:
886:
887:
888:
889:
890:
891:
892:
893:
894:
895:
896:
897:
898:
899:
900:
901:
902:
903:
904:
905:
906:
907:
908:
909:
910:
911:
912:
913:
914:
915:
916:
917:
918:
919:
920:
921:
922:
923:
924:
925:
926:
927:
928:
929:
930:
931:
932:
933:
934:
935:
936:
937:
938:
939:
940:
941:
942:
943:
944:
945:
946:
947:
948:
949:
950:
951:
952:
953:
954:
955:
956:
957:
958:
959:
960:
961:
962:
963:
964:
965:
966:
967:
968:
969:
970:
971:
972:
973:
974:
975:
976:
977:
978:
979:
980:
981:
982:
983:
984:
985:
986:
987:
988:
989:
990:
991:
992:
993:
994:
995:
996:
997:
998:
999:
1000:

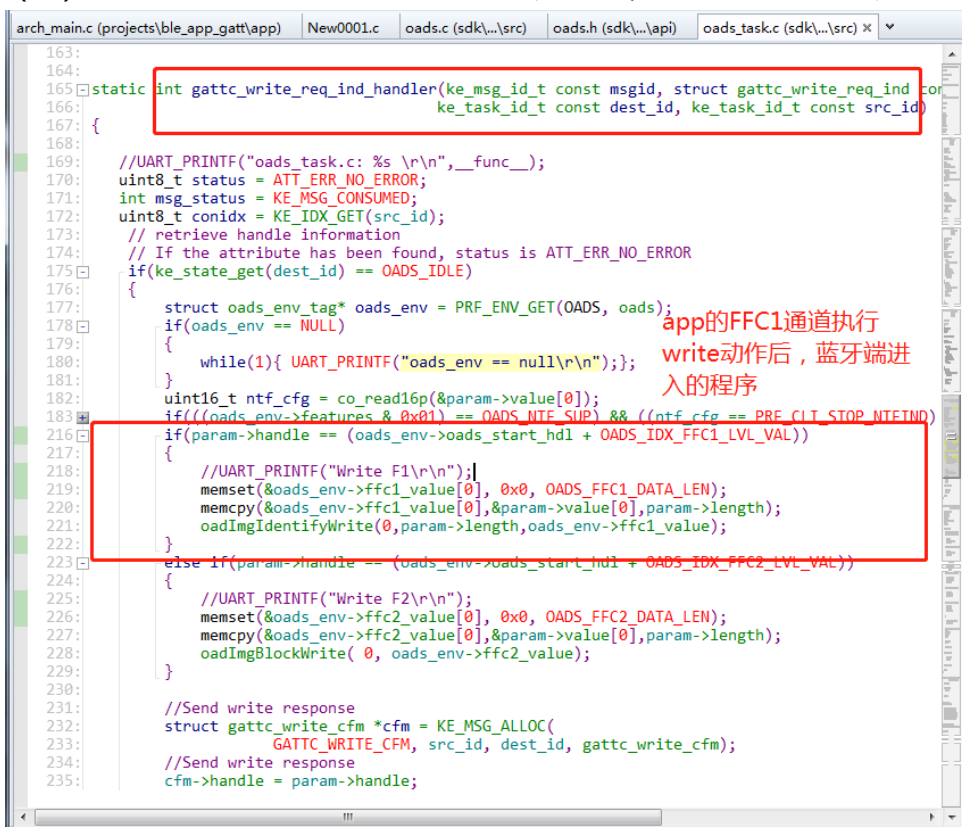
```

- (4) 不同升级模式的APP显示信息：

UUID	OTA	Other Files
Device Ver. :	1111	Dev. Rom Ver. : 0005
OTA File Ver. :	1111	File Rom Ver. : 0004
<div>ver版本号</div> <div>idle</div> <div>rom版本号</div>		
<div>bk3435_ble_app_stack_oad.bin</div> <div><input type="checkbox"/> bk3435_ble_app_oad.bin</div> <div><input type="checkbox"/> 部分OTA.bin</div> <div><input type="checkbox"/> 全部OTA.bin</div> <div><input type="checkbox"/> bk3432_ble_app_app.bin</div> <div><input type="checkbox"/> bk3435_ble_app_oad_on write.bin</div> <div><input type="checkbox"/> bk3435_ble_app_oad_noread onwrite.bin</div> <div><input type="checkbox"/> bk3435_ble_app_merge_crc.bin</div> <div><input checked="" type="checkbox"/> bk3435_ble_app_stack_oad.bin</div>		
<div>全部升级</div> <div>Full OTA</div>		
<div>bk3435_ble_app_oad_on write.bin</div> <div><input type="checkbox"/> bk3435_ble_app_oad.bin</div> <div><input type="checkbox"/> 部分OTA.bin</div> <div><input type="checkbox"/> 全部OTA.bin</div> <div><input type="checkbox"/> bk3432_ble_app_app.bin</div> <div><input checked="" type="checkbox"/> bk3435_ble_app_oad_on write.bin</div> <div><input type="checkbox"/> bk3435_ble_app_oad_noread onwrite.bin</div> <div><input type="checkbox"/> bk3435_ble_app_merge_crc.bin</div> <div><input type="checkbox"/> bk3435_ble_app_stack_oad.bin</div>		
<div>部分升级</div> <div>Partial OTA</div>		

2. 点击Full OTA或者Partial OTA按钮，APP通过FFC1的write通道将ver版本号和rom版本号下发给蓝牙，蓝牙对比后判断是全部升级还是部分升级。

(1) APP端在FFC1通道上执行write动作后，蓝牙端执行的程序。



```

163:
164:
165: static int gattc_write_req_ind_handler(ke_msg_id_t const msgid, struct gattc_write_req_ind const param,
166:                                       ke_task_id_t const dest_id, ke_task_id_t const src_id)
167: {
168:
169:     //UART_PRINT("oads_task.c: %s \r\n", __func__);
170:     uint8_t status = ATT_ERR_NO_ERROR;
171:     int msg_status = KE_MSG_CONSUMED;
172:     uint8_t conidx = KE_IDX_GET(src_id);
173:     // retrieve handle information
174:     // If the attribute has been found, status is ATT_ERR_NO_ERROR
175:     if(ke_state_get(dest_id) == OADS_IDLE)
176:     {
177:         struct oads_env_tag* oads_env = PRF_ENV_GET(OADS, oads);
178:         if(oads_env == NULL)
179:         {
180:             while(1){ UART_PRINT("oads_env == null\r\n");};
181:         }
182:         uint16_t ntf_cfg = co_read16p(&param->value[0]);
183:         if(((oads_env->features & 0x01) == OADS_NTF_SUP) && (ntf_cfg == PRF_CLT_STOP_NTFIND))
184:         {
185:             if(param->handle == (oads_env->oads_start_hdl + OADS_IDX_FFC1_LVL_VAL))
186:             {
187:                 //UART_PRINT("Write F1\r\n");
188:                 memset(&oads_env->ffc1_value[0], 0x0, OADS_FFC1_DATA_LEN);
189:                 memcpy(&oads_env->ffc1_value[0], &param->value[0], param->length);
190:                 oadImgIdentifyWrite(0, param->length, oads_env->ffc1_value);
191:             }
192:             else if(param->handle == (oads_env->oads_start_hdl + OADS_IDX_FFC2_LVL_VAL))
193:             {
194:                 //UART_PRINT("Write F2\r\n");
195:                 memset(&oads_env->ffc2_value[0], 0x0, OADS_FFC2_DATA_LEN);
196:                 memcpy(&oads_env->ffc2_value[0], &param->value[0], param->length);
197:                 oadImgBlockWrite(0, oads_env->ffc2_value);
198:             }
199:
200:             //Send write response
201:             struct gattc_write_cfm *cfm = KE_MSG_ALLOC(
202:                 GATT_WRITE_CFM, src_id, dest_id, gattc_write_cfm);
203:             //Send write response
204:             cfm->handle = param->handle;
205:         }
206:     }
207: }

```

(2) 选择部分升级还是全部升级：



```

212: void appm_update_param(struct gapc_conn_param *conn_param);
213: uint8_t oadImgIdentifyWrite(uint16_t connHandle, uint16_t length, uint8_t *pValue)
214: {
215:     UART_PRINT("oads.c: %s line: %d\r\n", __func__, __LINE__);
216:     img_hdr_t rxHdr;
217:     img_hdr_t imgHdr;
218:     rxHdr.ver = co_read16p(&(pValue[4]));
219:     UART_PRINT("rxHdr.ver = %x \r\n", rxHdr.ver);
220:     rxHdr.len = co_read16p(&(pValue[6]));
221:     UART_PRINT("rxHdr.len = %x \r\n", rxHdr.len);
222:     rxHdr.uid = co_read32p(&(pValue[8]));
223:     UART_PRINT("rxHdr.uid = %x \r\n", rxHdr.uid);
224:     rxHdr.rom_ver = co_read16p(&(pValue[14]));
225:     UART_PRINT("rxHdr.rom_ver = %x \r\n", rxHdr.rom_ver);
226:     if(rxHdr.uid == OAD_APP_PART_UID)
227:     {
228:         oad_firmware_type = 1;
229:         UART_PRINT("app part upgrade\r\n");
230:         //only app can be upgraded.
231:         flash_read(0, SEC_IMAGE_OAD_HEADER_APP_FADDR, sizeof(img_hdr_t), (uint8_t *)&imgHdr, NULL);
232:         UART_PRINT("imgHdr.rom_ver = %x \r\n", imgHdr.rom_ver);
233:         UART_PRINT("imgHdr.uid = %x \r\n", imgHdr.uid);
234:         UART_PRINT("imgHdr.ver = %x \r\n", imgHdr.ver);
235:         oadBlkTot = rxHdr.len / (OAD_BLOCK_SIZE / HAL_FLASH_WORD_SIZE);
236:         UART_PRINT("oadBlkTot = %x \r\n", oadBlkTot);
237:         if((imgHdr.ver != rxHdr.ver) && (oadBlkTot <= OAD_BLOCK_APP_MAX) && (oadBlkTot != 0) && (rxHdr.rom_ver != 0))
238:         {
239:             oad_uid_check_status = 1;
240:         }
241:     }
242:     else if(rxHdr.uid == OAD_APP_STACK_UID) //全部OTA：bin文件的版本信息
243:     {
244:         oad_firmware_type = 2;
245:         UART_PRINT("app and stack upgrade\r\n");
246:         //both app and stack can be upgraded.
247:         flash_read(0, SEC_IMAGE_OAD_HEADER_STACK_FADDR, sizeof(img_hdr_t), (uint8_t *)&imgHdr, NULL);
248:         UART_PRINT("imgHdr.rom_ver = %x \r\n", imgHdr.rom_ver);
249:         UART_PRINT("imgHdr.uid = %x \r\n", imgHdr.uid);
250:         UART_PRINT("imgHdr.ver = %x \r\n", imgHdr.ver);
251:         oadBlkTot = rxHdr.len / (OAD_BLOCK_SIZE / HAL_FLASH_WORD_SIZE);
252:         UART_PRINT("oadBlkTot = %x \r\n", oadBlkTot);
253:         if((rxHdr.rom_ver != imgHdr.rom_ver) && (oadBlkTot <= OAD_BLOCK_STACK_MAX) && (oadBlkTot != 0))
254:         {
255:             oad_uid_check_status = 1;
256:         }
257:     }
258: }

```

### (3) 将升级信息通过FFC2通道的notify回传给APP：

```
main.c (projects\ble_app_gatt\app) oads.c (sdk\...\src) * x oads.h (sdk\...\api) oads_task.c (sdk\...\src) Search Results
212: void appm_update_param(struct gapc_conn_param *conn_param);
213: uint8_t oadImgIdentifyWrite( uint16_t connHandle, uint16_t length, uint8_t *pValue )
214: {
215:     UART_PRINTF("oads.c:%s line:%d\r\n", __func__, __LINE__);
216:     img_hdr_t rxHdr;
217:     img_hdr_t imgHdr;
218:     rxHdr.ver = co_read16p(&pValue[4]);
219:     UART_PRINTF("rxHdr.ver. = %x \r\n", rxHdr.ver);
220:     rxHdr.len = co_read16p(&pValue[6]);
221:     UART_PRINTF("rxHdr.len. = %x \r\n", rxHdr.len);
222:     rxHdr.uid = co_read32p(&pValue[8]);
223:     UART_PRINTF("rxHdr.uid. = %x \r\n", rxHdr.uid);
224:     rxHdr.rom_ver = co_read16p(&pValue[14]);
225:     UART_PRINTF("rxHdr.rom_ver. = %x \r\n", rxHdr.rom_ver);
226:     if(rxHdr.uid == OAD_APP_PART_UID) ...
242:     else if(rxHdr.uid == OAD_APP_STACK_UID)//全部OTA：bin文件的版本信息 ...
259:     else ...
267:     if(oad_uid_check_status == 1)
268:     {
269:         oadBlkNum = 0;
270:         latency_disable_state = 1;//失能
271:         oad_uid_check_status = 0;
272:         //update oad connect parameter.
273:         struct gapc_conn_param param;
274:         param.intv_min = 12;
275:         param.intv_max = 14;
276:         param.latency = 0;
277:         param.time_out = 300;
278:         appm_update_param(&param);
279:         oadImgBlockReq(connHandle, 0);
280:     }
281:     else
282:     {
283:         oadImgIdentifyReq(connHandle, &imgHdr);
284:     }
285:     return ( 0x00 );//SUCCESS
286: }
287:
```

### (4) APP开始将bin文件写入0x44000地址，bin写入完成后，蓝牙重启，由boot将bin文件搬到指定位置，这部分用户不需要理会：

```
arch_main.c (projects\ble_app_gatt\app) oads.c (sdk\...\src) oads.h (sdk\...\api) oads_task.c (sdk\...\src) Search Results
163:
164:
165: static int gattc_write_req_ind_handler(ke_msg_id_t const msgid, struct gattc_write_req_ind const
166:                                         ke_task_id_t const dest_id, ke_task_id_t const src_id)
167: {
168:
169:     //UART_PRINTF("oads_task.c: %s \r\n", __func__);
170:     uint8_t status = ATT_ERR_NO_ERROR;
171:     int msg_status = KE_MSG_CONSUMED;
172:     uint8_t conidx = KE_IDX_GET(src_id);
173:     // retrieve handle information
174:     // If the attribute has been found, status is ATT_ERR_NO_ERROR
175:     if(ke_state_get(dest_id) == OADS_IDLE)
176:     {
177:         struct oads_env_tag* oads_env = PRF_ENV_GET(OADS, oads);
178:         if(oads_env == NULL)
179:         {
180:             while(1){ UART_PRINTF("oads_env == null\r\n");};
181:         }
182:         uint16_t ntf_cfg = co_read16p(&param->value[0]);
183:         if(((oads_env->features & 0x01) == OADS_NTF_SUP) && ((ntf_cfg == PRF_CLI_STOP_NTFIND) |
216:         if(param->handle == (oads_env->oads_start_hdl + OADS_IDX_FFC1_LVL_VAL))
217:         {
218:             //UART_PRINTF("Write F1\r\n");
219:             memset(&oads_env->ffc1_value[0], 0x0, OADS_FFC1_DATA_LEN);
220:             memcpy(&oads_env->ffc1_value[0], &param->value[0], param->length);
221:             oadImgIdentifyWrite(0, param->length, oads_env->ffc1_value);
222:         }
223:         else if(param->handle == (oads_env->oads_start_hdl + OADS_IDX_FFC2_LVL_VAL))
224:         {
225:             //UART_PRINTF("Write F2\r\n");
226:             memset(&oads_env->ffc2_value[0], 0x0, OADS_FFC2_DATA_LEN);
227:             memcpy(&oads_env->ffc2_value[0], &param->value[0], param->length);
228:             oadImgBlockWrite( 0, oads_env->ffc2_value);
229:         }
230:
231:         //Send write response
232:         struct gattc_write_cfm *cfm = KE_MSG_ALLOC(
233:             GATT_WRITE_CFM, src_id, dest_id, gattc_write_cfm);
234:         //Send write response
235:         cfm->handle = param->handle;
236:         cfm->status = status;
237:         ke_msg_send(cfm);
238:     }
239:     else if(ke_state_get(dest_id) == OADS_BUSY)
240:     {
241:         UART_PRINTF("OADS_BUSY\r\n");
242:     }
243: }
```

深圳市集贤科技有限公司